UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Universidad Internacional de La Rioja
Máster universitario en Seguridad Informática

Creación de Criptomoneda 'Token' E-USA de valor estable sobre la red Ethereum (SmartContract)

Trabajo Fin de Máster

Presentado por: Andrade Sánchez, Christian

Director/a: Sicilia Montalvo, Juan Antonio

Ciudad: Ambato / Ecuador
Fecha: Septiembre / 2018

Resumen

En la actualidad, la implementación de contratos inteligentes está creciendo a velocidades impresionantes, la nueva tecnología blockchain y la probada veracidad de los datos que ofrece, hace que cada vez, más instituciones privadas y públicas se adhieran a este tipo de tecnología con el objetivo de aprovechar todas las bondades que ofrecen.

Un problema puntual al cual nos enfrentamos al momento es la variación de precios en cuanto al valor de las criptomonedas utilizadas en los Smart Contracts, la denominada volatilidad, que afecta directamente en la tasa de adopción del usuario que está acostumbrado a pactar un valor y recibir el mismo valor pactado, cuando se utilizan criptomonedas para desplegar los contratos inteligentes los valores resultantes al cumplimiento de los mismos, suele ser otro valor diferente del pactado.

Para solucionar este problema, se propone la creación de un Token fijado al valor anclado de una moneda estable, como puede ser el Dólar USD, euro EUR, Libra esterlina GBP, etcétera, en éste trabajo se creará un Token con la paridad 1:1 con el valor del dólar y como la red que se utiliza es para el desarrollo es Ethereum, se lo ha denominado Ethereum-USA, USA, por ser el lugar donde se utiliza originalmente el Dólar Americano, y su símbolo será **“E-USA”**.

Se ha creado los Tokens a través de contratos inteligentes por medio de Solidity y se ha utilizado la red de prueba TestNet Ropsten.

El presente trabajo presenta una tónica particular y novedosa que justifica su creación, utilizando las diversas técnicas criptográficas que mantienen la seguridad de la red, el contrato inteligente que cree y utilice los Tokens valorados en $1 usd para el pago de servicios y proveedores por el valor de salarios cotizados en Ecuador a la fecha de implementación. El mismo que será pagado cada fin de mes. Y para seguridad y veracidad consta su respaldo en la blockchain.

Palabras Clave: Contratos Inteligentes, Blockchain, ERC20, Criptomonedas, Bitcoin, Ethereum, Solidity.

Abstract

Nota: Currently, the implementation of smart contracts is growing at impressive speeds, the new blockchain technology and the proven accuracy of the data it offers means that more and more private and public institutions adhere to this type of technology with the aim of take advantage of all the benefits they offer.

A specific problem that we face at the moment is the price variation in terms of the value of the cryptocurrencies used in the Smart Contracts, the so-called volatility, which directly affects the rate of adoption of the user who is accustomed to agree a value and receive the same agreed value, when cryptocurrencies are used to deploy intelligent contracts, the resulting values to comply with them, is usually a different value from the agreed one.

To solve this problem, we propose the creation of a Token fixed to the anchored value of a stable currency, such as Dollar USD, Euro EUR, GBP Pound Sterling, etc., in this work a token will be created with parity 1: 1 With the value of the dollar and how the network used for development is Ethereum, it has been named Ethereum-USA, USA, because it is the place where the American Dollar was originally used, and its symbol will be "E-USA".

The Tokens have been created through intelligent contracts through Solidity and the TestNet Ropsten test network has been used.

The present work presents a particular and novel tonic that justifies its creation, using the various cryptographic techniques that maintain the security of the network, the intelligent contract that creates and uses the Tokens valued at $ 1 usd for the payment of services and suppliers for the value of salaries quoted in Ecuador on the date of implementation. The same that will be paid each end of the month. And for security and veracity, its support in the blockchain.

Keywords: SmartContracts, Blockchain, Criptography, Cryptocurrency, Bitcoin, Ethereum, Solidity.

Índice de contenidos

Índice de tablas

Índice de Ilustraciones

1. Introducción

1.1 Justificación

¿Qué son los contratos inteligentes?

Adaptando a la idea de la publicación (Monax Industries, 2018), para empezar, los Contratos Inteligentes “SmartContracts” NO SON especialmente inteligentes, NI SON legalmente hablando, contratos.

En su lugar, se puede proponer un concepto menos subjetivo y más objetivo, son scripts dentro de la blockchain que representan promesas unilaterales para el cumplimiento y/o registro de transacciones en las que no intervienen seres humanos y una vez iniciado no hay manera de detener el script hasta su finalización.

Personalmente los llamaría “Pactos Inteligentes” ya que contrato en sí, implica o contiene una insinuación legal y jurídica a nivel global.

Estos scripts se compilan en códigos de operación de bajo nivel y se almacenan en una base de datos del blockchain dentro una dirección particular, que se crea cuando los contratos se implementan en el blockchain.

Cuando se envía una transacción a esa dirección, la máquina virtual distribuida en cada nodo completo de la red blockchain, ejecuta los códigos de operación del script utilizando los datos que se envían con la transacción.

Los contratos inteligentes son scripts modulares, repetibles y autónomos, que se pueden utilizar para crear aplicaciones individuales, para una comunidad, para un cliente, para una recompensa, o incluso solo por diversión.

Se pueden mezclar y combinar, y son fáciles de repetir, más bien como fichas de Lego combinados con plantillas preestablecidas.

Los contratos inteligentes en la red Bitcoin, existen pero son bastante limitados por la estructura de su blockchain ya que es una cadena de scripts específicos creados justamente para el registro inmutable de las transacciones de Bitcoin (BitcoinMagazine, s.f.), la creación de contratos inteligentes de cualquier tipo, dentro de la blockchain de Ethereum, se puede dar debido a que ésta blockchain es denominada de tipo Turing-Complete, es decir, es un equipo distribuido a nivel mundial que puede leer y ejecutar cualquier orden correctamente escrita

‘codificada’ por medio de la EVM ‘Ethereum Virtual Machine’ en una forma similar cómo funciona la JVM ‘Java Virtual Machine’.

Los contratos inteligentes pueden codificarse para reflejar cualquier tipo de lógica empresarial o de ingeniería basada en datos: desde acciones tan sencillas como la renovación de una publicación en un foro, hasta las más complejas, como la garantía de préstamos y contratos de futuros, hasta las altamente complejas como la priorización de pago en una nota estructurada (Pasten Lugo, 2012).

Las relaciones y obligaciones que son 'contractualmente inteligentes' se benefician de la lógica de seguridad blockchain y también del aumento en la verificabilidad que proporcionan las redes blockchain.

Básicamente se denomina contrato inteligente a una transacción programada en un solo documento, una DAAP ‘aplicación descentralizada’ es un conjunto de contratos inteligentes.

Si bien, en esencia, los contratos inteligentes son simplemente secuencias de comandos de software que no se diferencian de las que se pueden ejecutar en cualquier pila de aplicaciones, tienen una cualidad única para ellos que ningún otro script aleatorio de software tiene: CERTEZA.

Los contratos inteligentes tienen visibilidad a través de la cadena de bloques en la que viven, es decir, si alguien tiene acceso para leer el blockchain, tendrá acceso para ver el script compilado.

Con este concepto se puede asegurar que cierta secuencia de comandos funcionará de la misma manera sobre la blockchain, pero tal vez responda de otra manera en servidores que no se pueden controlar.

Los contratos inteligentes nos permiten la transferencia de valores y datos, en general (dinero, propiedades, acciones, historial médico, propiedad intelectual, etc.)

Recapitulando, una empresa en toda su vida, en promedio, mantiene relaciones comerciales con alrededor de una docena de empresas anexas, muy diversas, la forma en que los departamentos de T. I., puedan crear acuerdos, controlar, actualizar, consultar y monitorear toda la gestión que involucran los negocios, podría ser muy compleja al tener que desarrollar una cantidad similar de sistemas compatibles que sean accesibles desde todas las entidades involucradas, es aquí donde la certeza y fiabilidad de los contratos inteligentes y la blockchain entran en el juego, basados en la transparencia e inmutabilidad de la cadena de bloques, evitando el tener que desarrollar doce sistemas diferentes, con la consecuente mejora de tiempos, optimización de recursos y control integral de los proyectos.

Al utilizar sistemas centralizados e independientes, se puede dar el caso malintencionado de modificar los scripts o insertar funciones a discreción, durante o después de haber sido realizados, con los contratos inteligentes, luego de haber configurado los parámetros necesarios, este tipo de contrato es de cumplimiento unilateral, no se puede detener.

La dificultad que presenta la introducción generalizada de los contratos inteligentes, es la adopción masiva de ésta herramienta, el desconocimiento, y el manejo de las criptomonedas en general, por lo cual se propone una piloto experimental para la creación y correcta utilización de los contratos inteligentes sobre la red Ethereum.

Según (Topini, 2017) a nivel mundial se estima que el 0.07% de las personas utilizan Bitcoin, como se explica en la siguiente tabla extraída de un estudio realizado desde el 2009 hasta el 2015 donde también se destaca que casi 7 millones de personas utilizan Bitcoin al 2017.

Tabla 1 Usuarios estimados de Bitcoin en el mundo

País	Usuarios	Población
Alemania	234,579	80,640,000
Argentina	2,879	41,350,000
Australia	94,224	23,105,000
Brasil	52,976	195,632,000
Canadá	146,919	35,236,000
China	340,195	1,357,379,000
España	56,729	46,958,000
EEUU	1,073,050	316,102,000
Finlandia	29,665	5,436,000
Francia	65,933	63,820,000
Holanda	85,492	16,795,000
India	28,395	1,257,476,000
Italia	56,317	59,789,000
Polonia	88,905	38,548,000
Reino Unido	208,203	64,231,000
Rumania	26,158	19,858,000
Rusia	193,514	143,455,000
Suecia	51,936	9,595,000
Suiza	25,735	8,075,000
Ucrania	5,953	45,461,000
TOTAL	**2,867,757**	**3,828,941,000**
%	**0.07%**	**100%**

Adaptación de (Topini, 2017)

En el sitio web Bitcoin.com (Shares, 2016) comenta que el 39.9% de los usuarios de Bitcoin son de Europa, el 86.9% de los usuarios son masculinos, y el 32.8% se encuentran en el rango de edad entre 25 – 34 años.

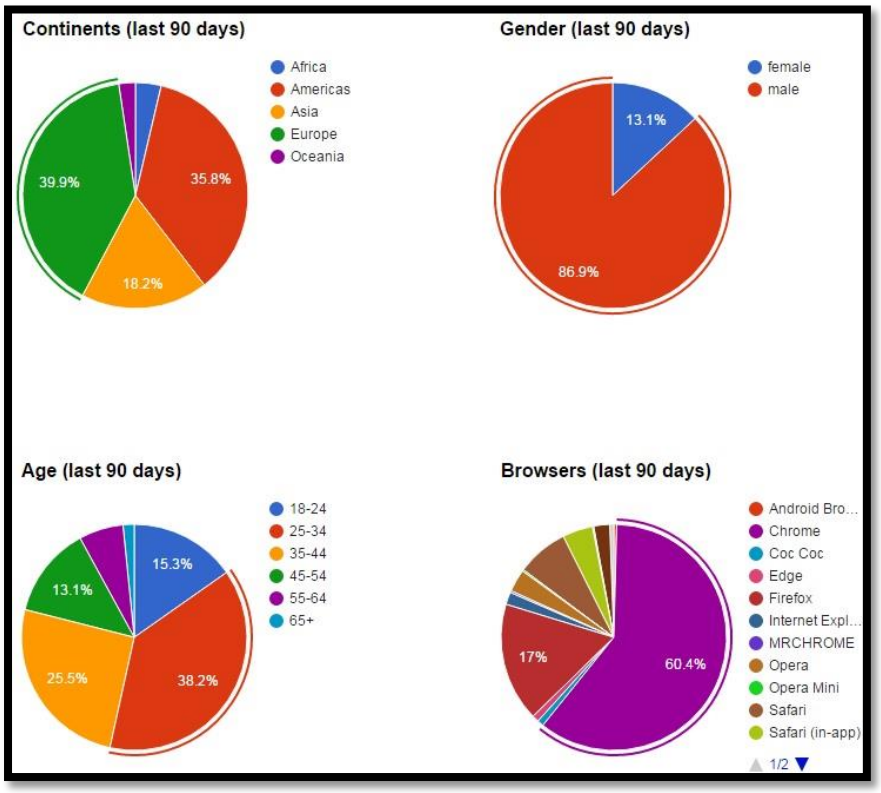


Ilustración 1 Demografía de Bitcoin (Shares, 2016)

1.2 Planteamiento del trabajo

La presente tecnología disruptiva que pretende 'cambiar el sistema', se presenta esquiva a las personas, puesto que en los contratos inteligentes se manejan criptomonedas que no son de manejo público, la mayor cantidad de posibles usuarios en la actualidad no tienen idea, ni el conocimiento de cómo crear una cuenta, cómo ingresar en este medio, cómo confirmar si las transacciones fueron efectuadas, como saber si una criptomoneda que reciben es verídica o es una estafa, ya que muchos han caído en redes de estafa, pirámides y el scam, no saben cómo proteger sus claves privadas tampoco lo qué éstas representan, en fin, al intentar dar a conocer el presente trabajo se pretende abarcar la mayoría de audiencia posible para poder evangelizar más el uso y las bondades de las criptomonedas, blockchain y contratos inteligentes, y todo el ecosistema de seguridad que las rodea.

Se plantea solucionar el problema al crear una un piloto experimental para la creación de una criptomoneda o Token basado en un contrato inteligente sobre la red Ethereum. Que sirva de base para personas con conocimientos altos, medios y bajos de programación, puedan llevar a cabo la utilización de los contratos inteligentes utilizando las redes criptográficas existentes como son Ethereum (tema de estudio) XEM, Lisk, Cardano, entre las más conocidas, además, pueden crear su propia red de criptomoneda o blockchain privada utilizando la red principal de IBM (IBM Blockchain) o con la plataforma HyperLedger, que es una asociación de varias empresas entre ellas se encuentra como fundamental Linux Foundation.

Entre los casos de uso que se puede proponer, la creación de una DAAP basada en contratos inteligentes la cual permita, realizar votaciones (sufragio), y como éstos datos se encuentran grabados en la blockchain no podrán ser adulterados. Otro ejemplo es que se pueda realizar la compra / venta de productos físicos mediante un contrato inteligente que tenga integrado un contrato de SCROW que permita la generación de confianza en los participantes y se puedan realizar compras / ventas con una reducción intrínseca de estafas y robos.

Por último proponer la creación de una criptomoneda “Token ERC20” basada en Ethereum y el mismo se encuentre anclado al valor del USD dólar para la realización de contratos inteligentes cuya fluctuación no sea volátil como es el caso de las criptomonedas, siendo éste problema uno de los impedimentos que hacen que esta tecnología pueda ser adoptada de manera general.

1.3 Estructura de la memoria

En el capítulo II se habla sobre el estudio del estado del arte, en el cual se analiza toda la bibliografía que interviene en el tema actual, que es prácticamente la introducción al Bitcoin, criptomonedas, la criptografía que se emplea, la evolución Ethereum, HyperLedger y se analiza los trabajos realizados por diferentes autores sobre la misma temática y se justifica la realización del presente trabajo.

En el capítulo III se realiza el Planteamiento de la Investigación, los experimentos para desarrollar el presente tema con la creación de un contrato inteligente que pueda a futuro administrar la renta de un vehículo inteligente que posea IoT integrado.

En el capítulo IV constan las pruebas y procedimientos efectuados en relación a la utilización de los contratos inteligentes y su seguridad, mostrando la creación de contratos inteligentes y desarrollando la creación de una moneda virtual.

En las conclusiones se realizó un análisis y valoración de los resultados obtenidos y las posibles líneas futuras de investigación sobre la forma en que las criptomonedas y los contratos inteligentes pueden mejorar la situación de la automatización de ejecuciones, la economía y el internet del valor.

2. Contexto y estado del arte

Para entender toda la terminología utilizada en este documento nos podemos referir al apartado **"Glosario de Términos"** detallado al final del documento en la sección de Anexos en el Anexo 1.

El Bitcoin y las "Alt Coins" -como se las conoce a las criptomonedas y proyectos que se crearon a continuación-, se encuentran en un virtual apogeo, pretendiendo una de ellas Ethereum, ser la moneda que comande el futuro de las criptomonedas, por la versatilidad de su cadena de bloques que permite, incluso programación avanzada sobre su Blockchain utilizándola como plataforma para los contratos inteligentes (Smart Contracts) y brindando la infraestructura y seguridad para muchos lanzamientos de nuevos proyectos ICO (Initial Coin Offering) y CrowdFounding (Recaudación de fondos para proyectos –Mecenazgo y Micro mecenazgo).

El Bitcoin no es solo una criptomoneda, es un protocolo de envío y recepción de mensajes opensource.

Bitcoin nace en 2009 gracias al paper o libro blanco "Bitcoin a peer to peer cash system" 'Bitcoin un sistema monetario persona a persona' escrito por el seudónimo "Satoshi Nakamoto" (Nakamoto, 2008), como resultado al rechazo al actual sistema monetario, que justificaba los salvatajes bancarios con el dinero del pueblo, creando un nuevo sistema de pagos electrónico que funcionaria en la red de internet únicamente, el cual no funcionaría de manera física en monedas o papel impreso, que tendría su base en el sistema peer to peer directamente entre usuarios, incluyendo un método de recompensas por colaborar con el ecosistema, sin intermediarios o terceros de confianza, consolidado sobre un sistema criptográfico de curva elíptica ECDSA de una sola vía creado a partir de firmas digitales llamado cadena de bloques, que utiliza un procedimiento de algoritmos matemáticos basados en funciones hash para la confirmación y almacenamiento inmutable de las transacciones y cambios de posesión. Este sistema generaría 21 millones de unidades de Bitcoin, y cada unidad tiene un sistema fraccionario de 8 decimales, siendo la unidad más pequeña denominada "Satoshi".

¿Cómo funciona Bitcoin técnicamente?

Para dar una idea de lo que realmente significa tener, enviar, minar Bitcoin, primeramente puedo exponer que significa Bitcoin, en su esencia es un archivo digital

> que registra los movimientos y los valores en un *"cuasi"* libro contable 'ledger' distribuido en la red de nodos Bitcoin a nivel mundial.
>
> Para enviar Bitcoin se anuncia a la red que una cantidad determinada de la cuenta de origen se debe restar y se la debe sumar a la cuenta de destino, entonces todos los nodos que comprenden la red Bitcoin actualizan su ledger, a lo que se le añade un poco de seguridad matemática (criptografía) basada en claves públicas y privadas para que estos registros se vuelvan inmutables, se puede ver en profundidad la teoría sobre el ECDSA algoritmo de firma digital de curva elíptica y mathematical trapdoor function.
>
> Bitcoin promete muchas ideas interesantes como el anonimato, costos bajos por transferencias de dinero a nivel mundial, independencia del gobierno y terceros de confianza, y también posee retos que debe superar como son la dificultad de convertir el Bitcoin en dinero de curso legal, el uso para operaciones fuera de la ley, la minería utiliza gran cantidad de energía eléctrica (CuriousInventor, 2013)

Las Criptomonedas ofrecen la versatilidad de pagos y transferencias, sin importar el lugar de origen, ni destino, sin intermediarios como bancos u otras instituciones financieras, incluso evitando el control de los gobiernos, cumpliendo con el objetivo de la descentralización utilizan un sistema peer to peer (punto a punto o P2P), pueden llegar a cualquier parte del mundo, el único requisito es tener acceso a internet.

Por supuesto nunca faltaran los cibercriminales que están atentos para poder infiltrarse en las transacciones y cometer delitos como hurtos, suplantación de identidad, estafas, entre otros, utilizando principalmente las técnicas más conocidas como son el Phishing, Scam, Ingeniería Social, entre las más utilizadas.

Técnicas de fuerza bruta o ruptura de contraseñas es una opción muy complicada, por no decir imposible, por cuanto los algoritmos que utilizan las criptomonedas para generar sus claves públicas y privadas son muy robustos.

El sistema en el que conviven las criptomonedas en un ecosistema llamado Blockchain o cadena de bloques que por su implementación es prácticamente imposible de vulnerar o realizar actos de corrupción. La Blockchain es un sistema descentralizado que, poniendo el ejemplo de Bitcoin, involucra más de 8000 terminales (nodos) alrededor del mundo.

Cada criptomoneda tiene su propia Blockchain, cada una tiene sus propias características de fortaleza, versatilidad y uso, que en su momento serán explicadas en el presente trabajo.

2.1. Criptomonedas

El término Criptomoneda (Cryptocurrency) viene de la unión de las palabras que hacen referencia a su nombre: cripto (criptografía) monedas (dinero).

Son valores digitales que se generan a partir de su Blockchain como un medio de monetización o forma de pago para cumplir contratos inteligentes, que a medida que se las utiliza y son aceptadas en más lugares, por la oferta y la demanda crecen en su valor.

Las criptomonedas tienen una tendencia Deflacionaria, al contrario que el dinero FIAT que se lo denomina así (dólar, euro, yen, libra, etc.) tienen tendencia inflacionaria y las transferencias internacionales dependen de un código SWIFT (Gil, 2015), que también fue Hackeado en Abril del 2016 como lo explica en el sitio web (Angulo, 2016), el mismo que se maneja a nivel mundial en la intermediación entre bancos, y puede llegar a tener costos y tiempos muy altos.

El sistema Bitcoin se encuentra organizado de tal manera que para aproximadamente en el año 2040 existirá una cantidad máxima del 99% de los 21 millones de monedas en total que habrá en circulación, su algoritmo ajusta la dificultad de cálculo para que un bloque sea encontrado cada 10 minutos, y el premio o recompensa por encontrar cada bloque disminuirá a la mitad cada 4 años, éste evento se denomina "***halving***":

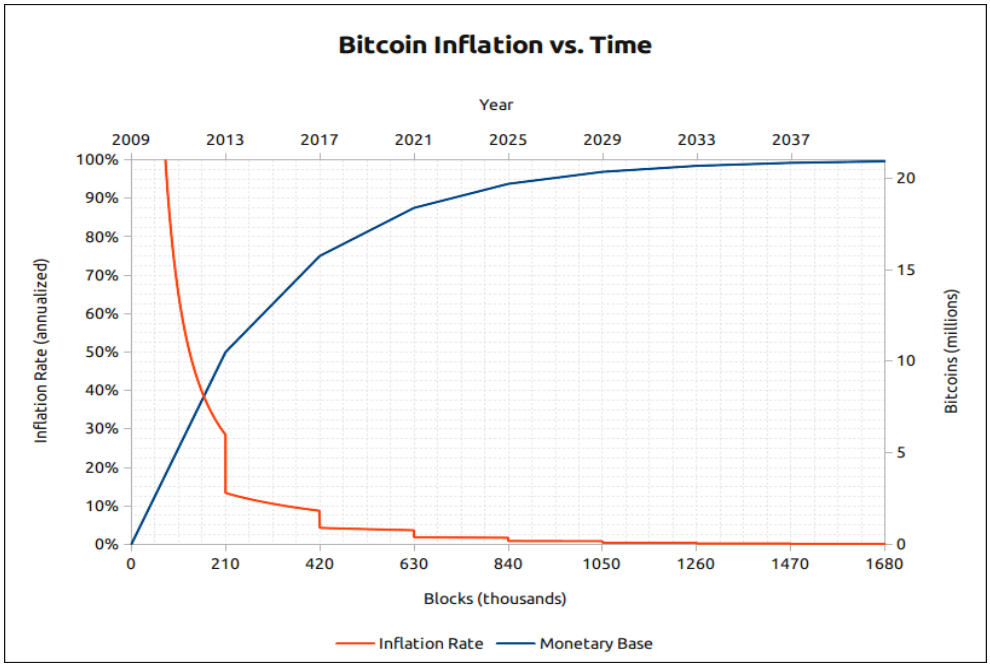


Ilustración 2 Inflación de Bitcoin vs Tiempo(Whitlock, 2012)

Tabla 2 Cantidad circulante Bitcoin

AÑO	RECOMPENSA BITCOINS	CANTIDAD CIRCULANTE CADA 4 AÑOS	CANTIDAD CIRCULANTE ACUMULADA
2008	50.00000000	10,630,000.00	10,630,000.00
2012	25.00000000	5,184,000.00	15,814,000.00
2016	12.50000000	2,592,000.00	18,406,000.00
2020	6.25000000	1,296,000.00	19,702,000.00
2024	3.12500000	648,000.00	20,350,000.00
2028	1.56250000	324,000.00	20,674,000.00
2032	0.78125000	162,000.00	20,836,000.00
2036	0.39062500	81,000.00	20,917,000.00
2040	0.19531250	40,500.00	20,957,500.00
2044	0.09765625	20,250.00	20,977,750.00
2048	0.04882813	10,125.00	20,987,875.00
2052	0.02441406	5,062.50	20,992,937.50
2056	0.01220703	2,531.25	20,995,468.75
2060	0.00610352	1,265.63	20,996,734.38
2064	0.00305176	632.81	20,997,367.19
2068	0.00152588	316.41	20,997,683.59
2072	0.00076294	158.20	20,997,841.80
2076	0.00038147	79.10	20,997,920.90
2080	0.00019073	39.55	20,997,960.45
2084	0.00009537	19.78	20,997,980.22
2088	0.00004768	9.89	20,997,990.11
2092	0.00002384	4.94	20,997,995.06
2096	0.00001192	2.47	20,997,997.53
2100	0.00000596	1.24	20,997,998.76
2104	0.00000298	0.62	20,997,999.38
2108	0.00000149	0.31	20,997,999.69
2112	0.00000075	0.15	20,997,999.85
2116	0.00000037	0.08	20,997,999.92
2120	0.00000019	0.04	20,997,999.96
2124	0.00000009	0.02	20,997,999.98
2128	0.00000005	0.01	20,997,999.99
2132	0.00000002	0.00	20,998,000.00
2136	0.00000001	0.00	20,998,000.00
2140	0.00000001	0.00	20,998,000.00

En sus inicios el Bitcoin obtuvo su valor al realizar su primera transacción por un bien tangible, se ofreció 10.000 BTC por 2 pizzas como nos comenta (Puigvert, 2016) en la conocida página de noticias CRIPTONOTICIAS, es entonces cuando el BTC empezó a ganar valor, y con el pasar del tiempo su demanda creció y su valor se incrementó a pasos agigantados, dicha transacción en el 2010 fue valorada en más o menos $30 USD, en la actualidad la misma transacción tendría un costo aproximado de $67.000.000 USD, como lo demuestra la siguiente tabla:

Tabla 3 Valor promedio anual Bitcoin

AÑO	VALOR (USD)	
2009	$	0.000074
2010	$	0.030000
2011	$	0.410000
2012	$	5.190000
2013	$	105.000000
2014	$	490.000000
2015	$	226.000000
2016	$	638.000000
2017	$	3,550.000000
2018	$	7,800.000000

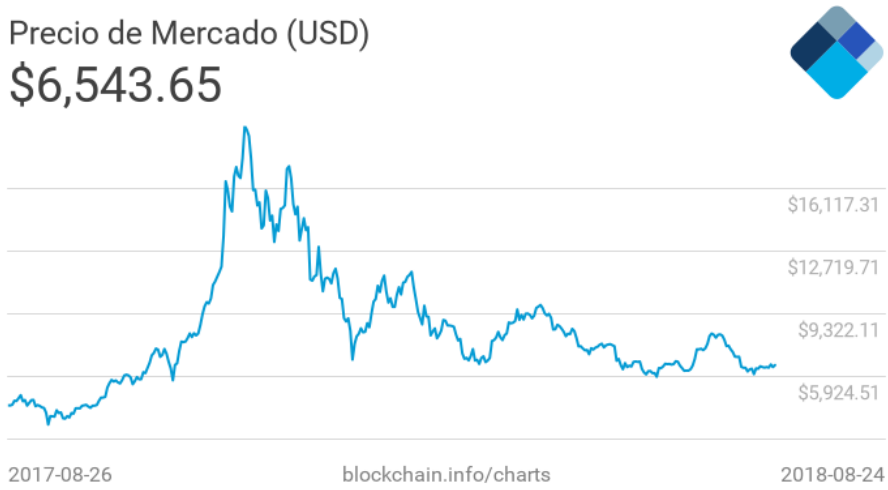


Ilustración 3 Bitcoin Precio de Mercado en USD (Blockchain.com, 2018)

La primera criptomoneda es Bitcoin creada en el 2009, seguida de cerca por proyectos como Ethereum, Ripple, BitcoinCash, Miota, Litecoin, Monero, Zcash, entre otros, todas estas criptomonedas o criptodivisas son fácilmente intercambiables en Exchanges o centros de intercambio de criptomonedas.

Las criptomonedas más conocidas en la actualidad son:

Tabla 4 Criptomonedas más conocidas

#	NOMBRE	ABREV.	ALGORITMO	SISTEMA	PRECIO	MINABLE
1	Bitcoin	BTC	SHA-256	POW	$ 2,957.1000	SI
2	Ethereum	ETH	Dagger-Hashimoto	POW	$ 364.0500	SI
3	Ripple	XRP	ECDSA	-----	$ 0.2704	NO
4	Ethereum Classic	ETC	Dagger-Hashimoto	POS	$ 22.0800	SI
5	NEM	XEM	BLOCKCHAIN	POI	$ 0.2134	NO
6	Litecoin	LTC	SCRYPT	POW	$ 32.6100	SI
7	Dash	DASH	X11	POW	$ 194.4200	SI
8	BitShares	BTS	Transaction fee	DPOS	$ 0.3445	NO
9	Monero	XMR	CryptoNight	POW	$ 55.8300	SI
10	Bytecoin	BCN	CryptoNight	POW	$ 0.0034	SI
11	Waves	WAVES	Transaction fee	POS	$ 6.1100	NO
12	Steem	STEEM	SHA-256	POW	$ 2.2300	SI
13	Zcash	ZEC	EQUIHASH	POW	$ 322.2600	SI
14	Stellar Lumens	XLM	Transaction fee	POW	$ 0.0454	NO
15	Dogecoin	DOGE	SCRYPT	POW	$ 0.0035	SI

De la anterior tabla de Criptomonedas podemos detallar las columnas:

ABREV: Abreviatura o símbolo que utiliza la moneda (similar a un logotipo).

ALGORITMO: Algoritmo utilizado por cada moneda para encriptar la información y realizar los registros y revisiones en cada Blockchain.

SISTEMA: El sistema que utiliza para la minería o revisión de transacciones POW (Proof of Work) prueba de trabajo, POS (Proof of Stake) Prueba de pila, etc.

PRECIO: El precio basado en USD.

MINABLE: Si la moneda puede ser creada y obtenida a través del proceso de minería por trabajadores a nivel mundial con utilización de hardware y software.

Existen diversas maneras de obtener criptomonedas, comprando por medio de personas que poseen las criptomonedas, un sitio recomendado es www.localbitcoins.com, que tiene calificaciones y número de transacciones positivas y negativas, otra forma es haciendo trading o comercio entre criptomonedas (a modo de Forex) luego de obtener nuestras criptomonedas, las podemos comerciar en Exchanges con las técnicas de comprar cuando está barato y vendiendo caro, una tercera forma de conseguirlas es minando (utilizando tus equipos de cómputo, de muy alto poder, y uniéndote al ecosistema, verificando transacciones) o

conformando un nodo POS, y por último se puede obtener criptomonedas realizando venta de productos y servicios teniendo como medio de cobro las criptomonedas.

CUADRO COMPARATIVO CARACTERISTICAS DE CRIPTOMONEDAS

Tabla 5 Cuadro Comparativo Criptomonedas

Datos	BITCOIN	BITCOIN CASH	ETHEREUM	LITECOIN	DASH	MONERO
Nomenclatura	BTC	BCH	ETH / ETC	LTC	DASH	XMR
Logotipo						
Bloque Génesis	Ene / 2009	Ene / 2009	Jul / 2014	Oct / 2011	Ene / 2014	Abr / 2014
Algoritmo	SHA256	SHA256	ETHASH	SCRYPT	X11	CryptoNight
Minable	SI - POW	SI - POW	SI - POW	SI - POW	SI - POW	SI - POW
Tamaño de Bloque	1MB	8MB	AJUSTABLE	1 MB	3 MB	AJUSTABLE
Tiempo en encontrar un bloque	~ 10 minutos	~ 10 minutos	~ 16 segundos	~ 2:30 minutos	~ 2:30 minutos	~ 2 minutos
Monedas por Bloques Encontrados	12.5	12.5	5	25	1.80147602	6.7
Explorador de Blockchain	Blockchain / https://blockchain.info	https://blockchair.com/	Etherchain / https://etherchain.org - https://gastracker.io/	http://explorer.litecoin.net - http://ltc.blockr.io - https://live.blockcypher.com/ltc	https://chainradar.com - https://chainz.cryptoid.info/dash	http://moneroblocks.info/ - https://chainradar.com/xmr/blocks
Transacciones diarias	261,588	8,181	354,563	12,500	72,000	4,600
Confirmaciones mínimas por transacción	6	6	50 / 100	6	3	25
Media de tiempo en confirmación de transacciones	15 minutos	120 minutos	4 minutos	2.5 minutos	12	15 minutos
Dificultad	708,659,466,230	109,634,813,602	1,154,893,537,540,920	782,778	1,570,200	31,150,569,684
Curva de dificultad promedio	14 días	14 días	~ 16 segundos	3.5 días	~ 2:30 minutos	~ 2 minutos
Ajuste de dificultad	2016 bloques	2016 bloques	Dinámico cada bloque	2016 bloques	cada bloque	cada bloque
Ejemplo de número de Cuenta (Wallet) – Clave Pública –	1GjG1RTCHxzdjwjZd1YLVwdvjhQj9JoTf8	14nMF4X3WzSxe3bCDYcSUDdrKPsj6Ar8bg	0xe2928a7cb8165fe4e083fe93f481c38febf13396	LTGSmxRguq752mCvxTTUhGzSWHBxRqt4Fj	XohrU5PebgDCcFqNNU8GCdz5jYoe11yNnD	4BCeEPhodgPMbPWFN1dPwhWXdRX8q4mhhdZdA1dtSMLTLCEYvAj9QXjXAfF7CugEbmfBhgkqHbdgK9b2wKA6nqRZQCgvCDm
Monederos Conocidos	Bitcoin.com / XAPO / Jaxx / Exodus / Blockchain.org	Jaxx / Ledger/ Trezor	Jaxx / Exodus / MyEtherWallet	Jaxx / IOAFwALLET	Jaxx /	MyMonero / LightWallet / FreeWallet
Valoración en USD a la fecha	$3,900.00	$470.00	$245.25 / $17.28	$51.00	$315.00	$99.00
Unidades emitidas/totales	16,561,954 / 21,000,000	16,580,829 / 21,000,000	94,654,575 / no confirmado	52,998,532 / 82,000,000	7,567,484 / 18,500,000	15,088,000 / 18,446,744,073,709,551,616
Unidades Fraccionarias	cBTC, mBTC, uBTC, Satoshis	cBTC, mBTC, uBTC, Satoshis	finney, szabo, shannon, babbage,lovelace.	-	-	-
Web Site	www.bitcoin.org	bitcoincash.org	ethereum.org	litecoin.org	www.dash.org	getmonero.org
Nodos	8,778	1,100	1,200	16,710	6,450	16,200
Creador / Fundador	Satoshi Nakamoto	-	Vitalik Buterin	Charlie Lee	Evan Duffield	Riccardo Spagni

Adaptación de: (bitinfocharts@gmail.com, 2018) **-** (CoinWarz, 2018) **-** (coinmarketcap.com, 2018)

2.1.1. Saldos y transacciones Bitcoin

(Narayanan, y otros, 2016) Explican cómo se conforman los saldos de una cuenta Bitcoin y un ejemplo de una transacción, la misma que se encuentra conformado de la suma de todas las entradas confrontadas contra la suma de todas las salidas, si el saldo es positivo y suficiente para realizar la transferencia de valores, de la siguiente manera:

Ilustración 4 Entradas y Salidas de transacción Bitcoin (Narayanan, y otros, 2016)

Una transacción de Bitcoin se puede evidenciar de la siguiente manera:

En la siguiente ilustración tomada de (blockchain.info, 2017) se presenta de forma gráfica, cómo un bloque se diversifica en las transacciones que contiene.

En el consta el Hash de la transacción que contiene un número hexadecimal resultado de un algoritmo SHA-256 que se presenta en este formato:

1383ec8588bf76e1b17e7981e48e383f5a865bbd4543185c4c877ba127ca0dce

Posteriormente se encuentra todas las transacciones que contiene un bloque de máximo 1MB representado en el círculo principal del lado izquierdo donde se despliegan la cantidad total de las transacciones.

Dentro de las transacciones del bloque se encuentra la figura del círculo más grande, es en donde consta la transacción que contiene el mayor valor.

Los círculos presentados, tienen 2 colores, naranja y celeste.

El color naranja representa las transacciones que ya fueron utilizadas o gastadas y pueden desplegar sus nuevas transacciones con la trazabilidad de a donde fueron esos valores (transacciones hijas), dando click sobre ellas.

Las de círculo de color celeste, son las transacciones o entradas que aun contienen los valores y no tienen movimientos posteriores (hijos).

> “Los algoritmos para comprobar si un bloque es válido, expresado en este paradigma como sigue:
>
> Verifica si el bloque previo referenciado por el bloque existe y es válido.
>
> Verifica que la estampa de tiempo del bloque es mayor que la del bloque previo y menor que 2 horas en el futuro.
>
> Verifica que la prueba de trabajo en el bloque sea válida.
>
> Sea S[0] el estado al final del bloque previo.
>
> Suponiendo TX como la lista de transacciones del bloque con n transacciones. Para todo i en 0...n-1, implica que S[i+1] = APLICA(S[i+1], TX[i]) Si alguna aplicación devuelve un error, sale y devuelve falso.
>
> Devuelve true, y registra S[n] como el estado al final de este bloque” (James, 2018)

Como se puede observar en la siguiente Ilustración:

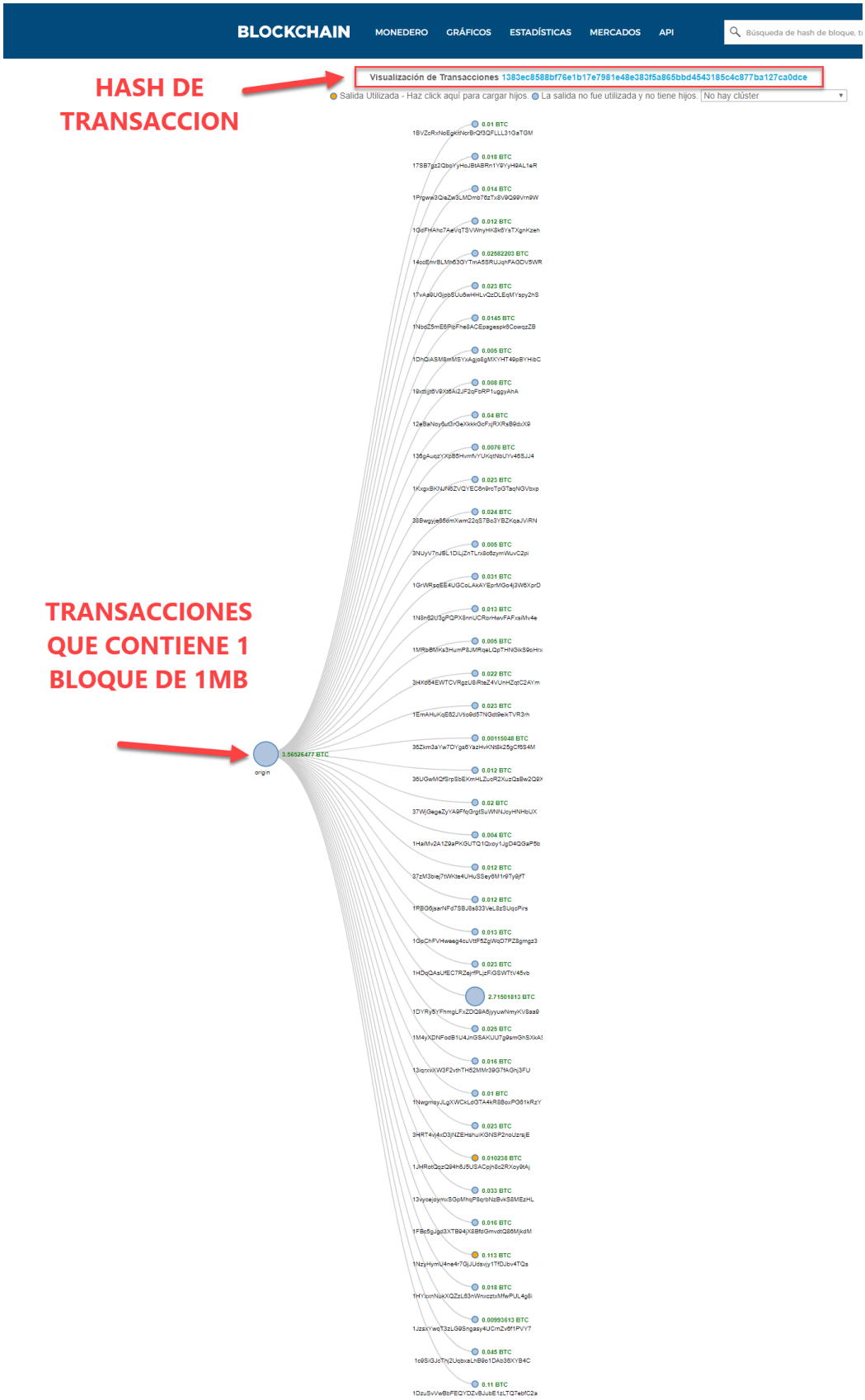


Ilustración 5 Transacciones de Bloque 1MB (blockchain.info, 2017)

Ataque por Maleabilidad de las transacciones

El tiempo que demora en confirmar una transacción Bitcoin se basa en una profundidad de 6 bloques, si tomamos en cuenta que cada bloque de Bitcoin se encuentra cada 10 minutos, una transacción realizada con 1 hora de antelación, se encuentra totalmente confirmada, una transacción realizada hace 5 minutos es altamente probable que pueda ser vulnerada.

“La maleabilidad de las transacciones es algo que se conoce desde el 2011. En términos más sencillos, es la posibilidad de modificar la identificación durante la pequeña ventana de tiempo que se inicia en el momento en que se autoriza la transacción y se cierra cuando es confirmada en la cadena de bloques. Esto es algo que no se puede corregir de un momento a otro. Por lo tanto, cualquier empresa que trata cotidianamente con transacciones Bitcoin y ha programado su propio software de cartera debe tenerlo en cuenta y prepararse de manera responsable, incluyendo en su software una manera de validar las identificaciones.” (Andresen, 2014)

Temas que son significativos en relación a las criptomonedas se detallan en la sección Anexos al final del documento dentro del Anexo 2.

2.2. Blockchain

La tecnología Blockchain es considerada la nueva revolución industrial, para enfocarnos en la grandeza de este hecho, lo comparamos con la creación de la tecnología de la información y comunicación creada alrededor de 5 siglos atrás, la Imprenta, el cambio que ofreció en su tiempo esta tecnología que es la que permitió que mucha gente pueda salir de la ignorancia y pasar de ser unos pocos los que tenían acceso a la lectura y la escritura, que siempre eran los más poderosos y millonarios, a la actual fuente de conocimiento y comunicación global, hoy los niños de 6 años ya saben leer y escribir, imaginarse que antes de la creación de la imprenta un niño de 6 años tuviese este conocimiento, es como esperar que en la actualidad niños de esa edad pudieran programar computadores.

Existen varios tipos de blockchain, **públicas**, en las que no existen restricciones para leer los datos y se pueden enviar transacciones para incluirlas en la blockchain, **privadas**, en las que para realizar transacciones y remitir datos a la blockchain está limitado a un grupo o listado predefinido, **permisionadas**, en el cual el proceso de las transacciones está sujeto a un listado o grupo de entidades conocidas y **no permisionadas**, que cualquier entidad puede realizar el proceso de transacciones.

Para hablar de la Blockchain, primeramente, se debe entender los tipos de sistemas informáticos que existen en la actualidad y por qué se diferencian de la cadena de bloques.

Centralizado, Descentralizado y Distribuido.

2.2.1. Centralizado:

Es el sistema fundamental y básico en el cual se utiliza la base de datos directamente de manera cliente – servidor, acompañado por planes de contingencia y de recuperación ante desastres junto con otros sistemas centralizados reflejados, como los sistemas bancarios, sistemas de control de ventas, sistemas contables de empresas, etc.

En este sistema si el servidor central es atacado, se cae el sistema.

Este contexto se puede calificar como de uno a muchos, de un rey a los súbditos, un presidente a sus soberanos, etc.

Es considerado el Pasado.

Ilustración 6 Sistema Centralizado

2.2.2. Descentralizado:

Es el sistema más utilizado en la actualidad, se basan en unidades centrales alrededor del mundo y sus bases de datos se encuentran separadas basándose en sectores o regiones para brindar un servicio más eficiente, entre las cuales, destacan aplicaciones como:

Uber (sistema independiente de gestión de taxis a nivel mundial), Airbnb (sistema de reservas de estadías para vacaciones), Facebook, Google, etc. Una relación de pocos a muchos. Como tener una radio o un canal de televisión, pocos emisores muchos oyentes/televidentes.

Es considerado el presente.

Ilustración 7 Sistema Descentralizado

2.2.3. Distribuido:

Es la tecnología que mantiene una copia de la base de datos actualizada automáticamente en cada uno de los participantes de la red, de tal modo que existe un sistema más confiable y transparente en cada uno de los movimientos que se realizan alrededor de este ecosistema, funciona cuando sus integrantes llegan a consensos y se aprueba el ingreso de un nuevo bloque a la cadena, además, posee una robustez y seguridad probada por el uso de la criptografía como su principal factor de confianza.

Este sistema es especialmente utilizado en la actualidad por las criptomonedas, incluyendo de a poco otras formas de uso como los contratos inteligentes, contabilidad distribuida, control de activos, control de cadenas de producción, mantenimiento de un único historial clínico de los pacientes, sistemas de elecciones políticas, transparencia y seguimiento de donaciones a ONG's, sistemas automatizados de facturación y pago inmediato de impuestos (llegaría el fin de las facturas falsas o prestadas, la facturación duplicada o la doble contabilidad), entre muchas otras.

En este sistema, la seguridad cuenta y mucho, en el caso de la disponibilidad se necesitaría tan solo un par de todos los nodos para seguir brindando el servicio, en la confidencialidad el

uso extremo de la criptografía, hace que el sistema sea un tanto lento en relación a otros sistemas, pero garantiza un altísimo nivel de seguridad, al hablar de la integridad de los datos, existe toda la transparencia desde el origen de los registros hasta el último actualizado, por ejemplo, se puede hacer consultas sobre todo el recorrido de una transacción.

Se lo define como muchos a muchos, todos para todos, todos contra todos, todos podemos expresarnos y llegar a muchas personas, ejemplo: IPFS, videos en streamig, etc., todos podemos o tenemos la capacidad de crear contenido y llegar a las personas.

Es considerado el Futuro.

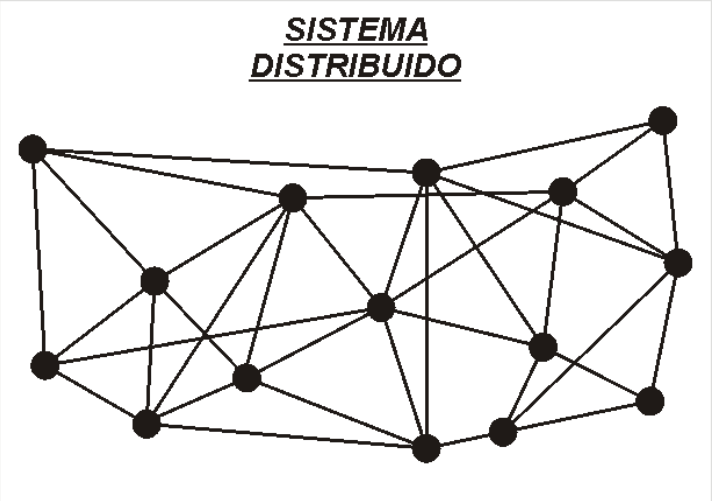


Ilustración 8 Sistema Distribuido

El resumen de una presentación de la empresa (SofttekTV, 2017) se explica cómo funciona Bitcoin de manera simplificada, que es la Blockchain, qué problemas puede resolver, dónde se está utilizando.

1. Inicia la transacción.
2. Se realiza una difusión de la transacción a toda la red Bitcoin.
3. Inicia el proceso de validación y verificación de las transacciones.
4. Una vez verificado los datos se genera un nuevo bloque.
5. Este bloque se distribuye a todo el sistema blockchain.
6. Se completa la transacción.

Blockchain es la tecnología que hace funcionar a Bitcoin, se considera una ***Base de Datos distribuida*** 'en una red de servidores', además se la considera ***inmutable, inviolable, incorruptible***, en la cual ***se guardan bloques*** de información que registran las monedas y las transacciones que se realizan con ellas y que ***no requiere de una autoridad central*** que actúe como intermediaria para validar estas transacciones, en resumen la confianza va a residir en el software.

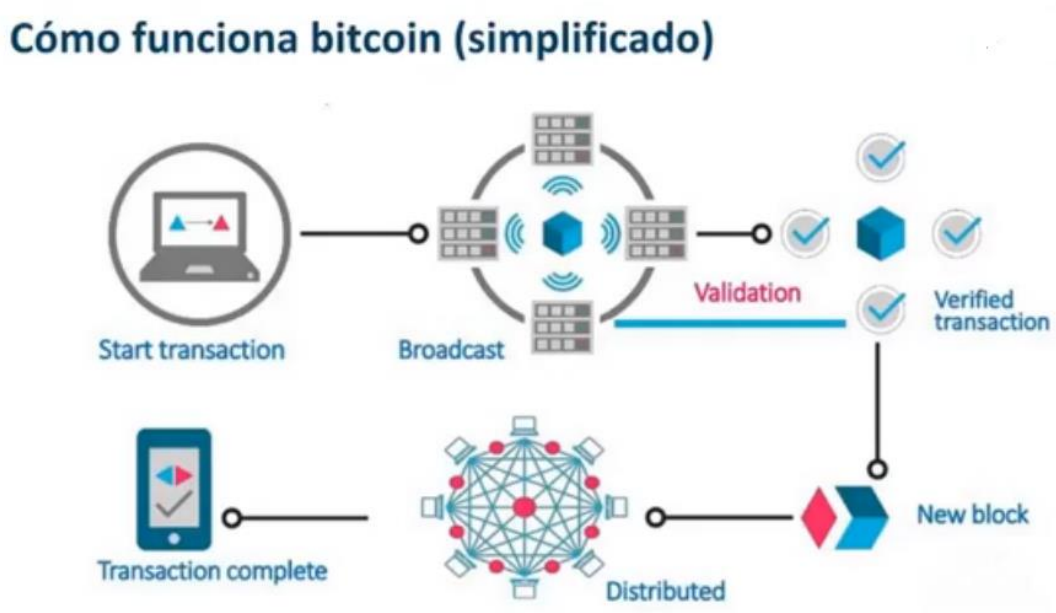


Ilustración 9 Cómo funciona Bitcoin 'simplificado' (SofttekTV, 2017)

Según (Narayanan, y otros, 2016) en la blockchain de Bitcoin el algoritmo criptográfico utilizado es SHA-256, el mismo que usa una compresión que toma una entrada de 768-bit y produce salidas de 256-bits, el tamaño del bloque es de 512 bits. Como se demuestra en la ilustración siguiente:

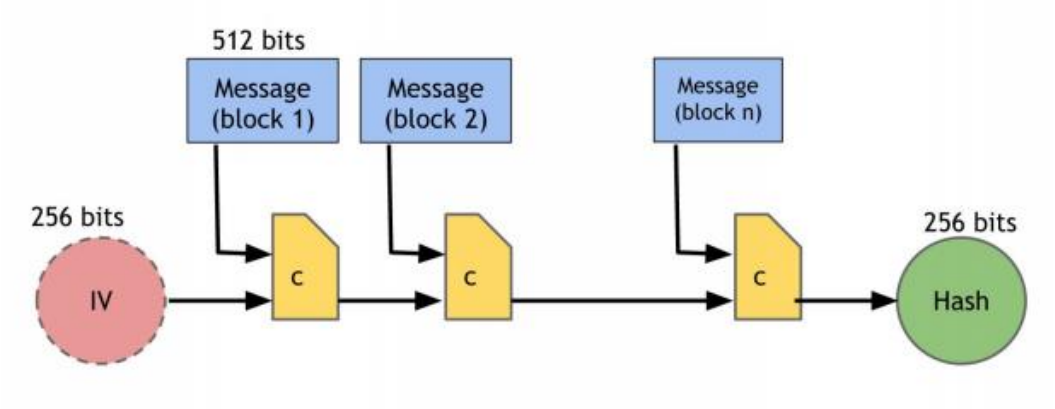


Ilustración 10 Función Hash 256 'simplificado' (Narayanan, y otros, 2016)

En la ilustración sobre la descripción de la cadena de bloques 'blockchain', (Narayanan, y otros, 2016) explican: Construimos una lista enlazada usando punteros de hash, y vamos a llamar a esta estructura una cadena de bloques, cada bloque se encuentra enlazada a la cabecera del bloque anterior, en el mismo encontramos, no solamente donde está ubicado el bloque anterior sino el resultado de su hash, y un resumen que nos permite verificar que su valor no ha cambiado.

Una cadena de bloques es una lista enlazada que se construye con punteros hash en lugar de solo punteros.

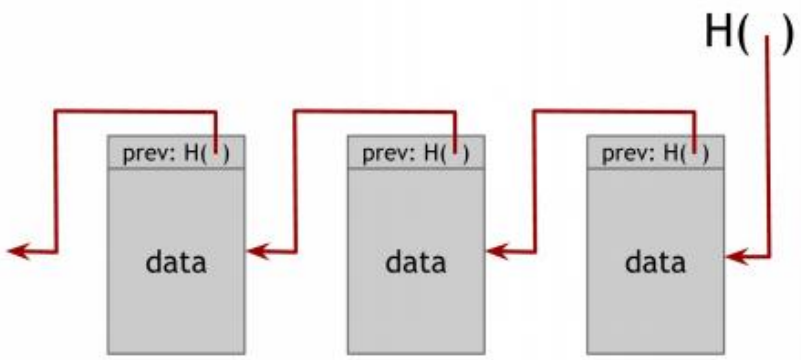


Ilustración 11 Descripción gráfica de la Cadena de Bloques (Narayanan, y otros, 2016)

2.2.4. Árboles de Merkle 'Merkle trees'

Según (Narayanan, y otros, 2016) nos brindan una explicación de los árboles de Merkle: Otra estructura de datos útil que podemos construir usando punteros hash es un árbol binario. Un árbol binario conformado de punteros hash es conocido como árbol de Merkle 'Merkle tree' nombrado así por su inventor Ralph Merkle. Supongamos que tenemos varios bloques que

contienen datos, estos bloques comprenden las hojas de nuestro árbol. Agrupamos estos bloques de datos en pares de dos, y luego para cada par creamos una estructura de datos que tiene dos punteros de hash, uno para cada uno de estos bloques. Estas estructuras de datos forman el siguiente nivel del árbol. A su vez los agrupamos en grupos de dos, y para cada par, creamos una nueva estructura de datos que contiene el hash de cada uno. Continuamos haciendo esto hasta que lleguemos a un solo bloque, la raíz del árbol.

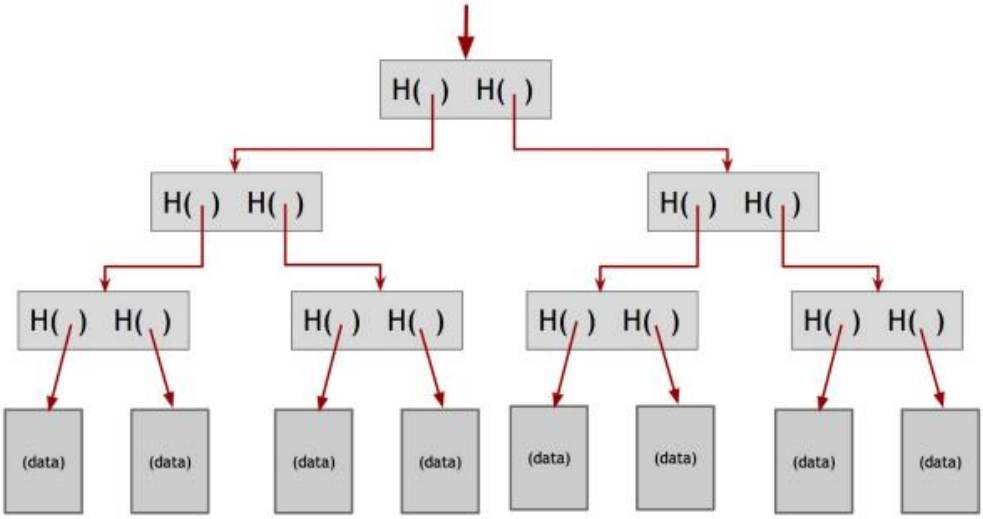


Ilustración 12 Árbol de Merkle (Narayanan, y otros, 2016)

2.3. HyperLedger

Según resumen de (Medina, 2018) HyperLedger es un proyecto open source, de software libre, que fue creado por la Linux Foundation en el año 2015, con la idea de permitir la utilización de la cadena de bloques más allá de las criptomonedas, desarrollando estándares herramientas y comunidades.

Este proyecto cuenta con más de 100 miembros, entre los principales podemos nombrar los siguientes:

Miembros centrados en la blockchain: Blockchain, ConsenSys, R3.

Compañías de tecnología: Cisco, Digital Asset Holdings, Fujitsu, Hitachi, IBM, Intel, NEC, NTT DATO, Red Hat, VMware.

Compañías financieras: ABN AMRO, ANZ Banco, BNY Mellon, Grupo CLS, Grupo CME, The Depository Trust & Clearing Corporation (DTCC), Grupo Deutsche Börse, J.P. Morgan, State Street, SWIFT, Wells Fargo.

Empresas varias: Accenture, Calastone, Créditos, Guardtime, IntellectEU, Nxt Fundation, Symbiont.

A partir de marzo del 2016 se unieron las siguientes empresas: Blockstream, Bloq, eVue Digital Labs, Gema, itBit, Milligan Socios, Montran Labs, Ribbit.me, TTequa Creek Holdings, Thomson Reuters, Beijing Digital Finance, Broadridge, Cloudsoft, Coinplug, Cuscal, Eurostep Holdings, Soramitsu, Skry, Belink Technologies, BitSe, INVeSHARE, MonetaGo, Moscow Exchange, Norbloc y Onchain

HyperLedger crea blockchain para ser usadas en un entorno privado, con el fin de que exista una comunicación entre empresas con todo el poder que brida una base de datos distribuida, inmutable y cifrada.

Según Brian Behlendorf director del proyecto HyperLedger, nunca creará su propia criptomoneda ya que la finalidad del proyecto es diferente.

El proyecto se divide en dos bloques marcados. El primero trata de los FrameWorks y el segundo de las herramientas para la utilización.

2.3.1. FrameWorks de HyperLedger

HyperLedger FABRIC que fue creado y donado por IMB que se utiliza para diseñar los entornos de comunicación entre empresas.

HyperLedger INDI es la que se ocupa de manejar todo lo concerniente con Identidad descentralizada.

HyperLedger IROHA está pensada para crear cadenas de bloques y operar en ellas, está enfocado hacia los dispositivos móviles, tiene bibliotecas para Android e iOS.

HyperLedger SAWTOOTH es desarrollado por Intel, su característica principal es que su consenso está basado en la prueba de duración de tiempo ‘proof of elapsed time’, y separa la capa de aplicación con la capa de negocio permitiendo programar en muchos más lenguajes y posee una característica importante como es la implementación de transacciones paralelas conocidas en programación como ‘multi hilos’.

HyperLedger BURROW utiliza la máquina virtual de Ethereum 'EVM' y su forma de comunicarse con las aplicaciones es mediante RPC, se programa de igual manera mediante solidity y utiliza toda la tecnología de Ethereum que existe en la actualidad.

2.3.2. Herramientas de HyperLedger

HyperLedger COMPOSER que permite crear los contratos inteligentes mediante JavaScript y brinda una API con las tareas más comunes que se realizan para poder obtener una programación más rápida. COMPOSER Convierte el chainCode de los contratos inteligentes programados en lenguaje GO a JavaScript.

COMPOSER es un conjunto de librerías o API's que permitirán conectar las aplicaciones desarrolladas con la blockchain de FABRIC

HyperLedger CELLO permite crear una blockchain bajo demanda, que se despliegue en la nube y sea como un BAAS 'Blockchain as a Service'.

HyperLedger EXPLORER que permite visualizar las operaciones de la blockchain, es una aplicación web que permite monitorear cada movimiento, como están los nodos, etc., es muy útil para realizar auditorías y funciona como un Dashboard o panel de control de todo el movimiento de la red.

HyperLedger QUILT que se pretende sea un protocolo de pagos entre las blockchain que existan que aún se encuentra en la fase BETA.

La característica principal de HyperLedger Fabric es permisionable, esto indica que puede conceder o retirar permisos, lo quiere decir que los nodos que pertenecen a esta red no necesariamente tienen acceso a toda la blockchain.

Al ser una red de tipo no anónima, se puede disminuir el nivel de confianza, sin la necesidad de aplicar el Proof of Work como en la red Bitcoin o Ethereum en la que se llega a gastar demasiados recursos, por el contrario los nodos existen con propósitos diferentes, tenemos nodos de tipo, se manejan diferentes tipos de consenso, depende del que escoja el administrador, existen canales internos dentro de la blockchain, lo que hace que no todos los nodos se comuniquen entre ellos.

HyperLedger tiene tres tipos de nodos, el primer tipo es el nodo es el que se encarga de colocar los bloques dentro de la blockchain 'COMMITER', otros nodos validan la información antes de que pertenezca al bloque 'ENDORSER', y finalmente los nodos que organizan los

bloques ‘CONSENTERS’, en este sistema ya no existe la minería, no existe la recompensa por ejecutar las transacciones, cada nodo realiza su trabajo, las transacciones se realizan sin la necesidad de tener asociada una criptomoneda.

HyperLedger posee una base de datos adicional llamada WorkState que guarda el estado actual de la blockchain y sirve para hacer consultas rápidas sin necesidad de recorrer cada uno de los bloques.

HyperLedger Fabric tiene un SDK para trabajar en Java y en NodeJS, es de código abierto y debido a la cantidad de empresas que trabajan con esta tecnología, empiezan a crearse formas de estandarización.

Los contratos inteligentes dentro de HyperLedger se denominan ChainCode.

La utilización de HyperLedger COMPOSE permite utilizar plantillas y programación de manera más gráfica, digamos que de forma mixta, no todo es código, ya que se puede utilizar en un complemento de programación llamad PLAYGROUND.

Para realizar los contratos inteligentes en HyperLedger se necesita instalar Docker Engine, Docker Compose, Microsoft Visual Studio, además tener conocimientos en Angular, C#, HTML5, JSON, manejo de GitHub y un grado más elevado de conocimientos en programación.

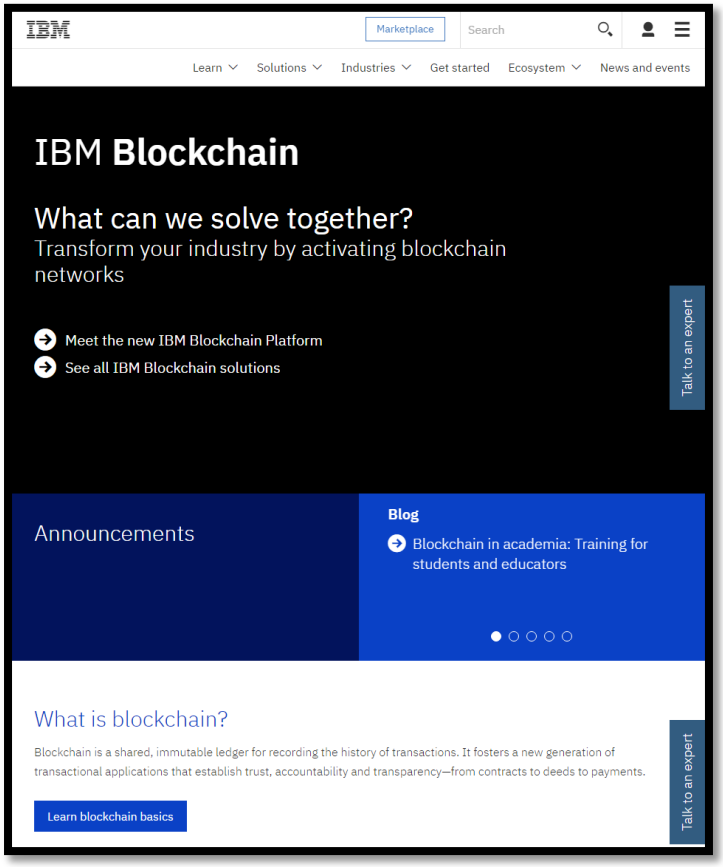


Ilustración 13 Blockchain de IBM (Juniper Research, 2017)

HyperLedger es un proyecto en el cual, en base a programación, se puede realizar la creación de Blockchains independientes para cualquier uso, brindando las herramientas necesarias con el fin de que cualquier persona, empresa o institución gubernamental pueda realizar sus pruebas y a futuro realice sus proyectos individuales en base a la tecnología Blockchain.

Como podemos observar en (Hyperledger , 2016) El producto HyperLedger Fabric de IBM ofrece los siguientes servicios:

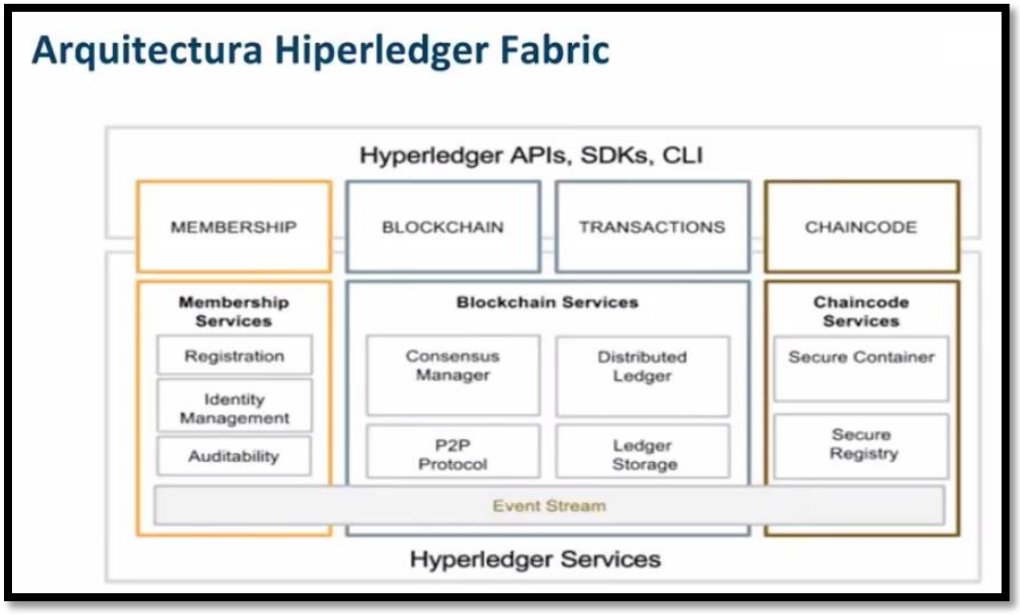


Ilustración 14 Arquitectura Hyperledger Fabric IBM (Hyperledger , 2016)

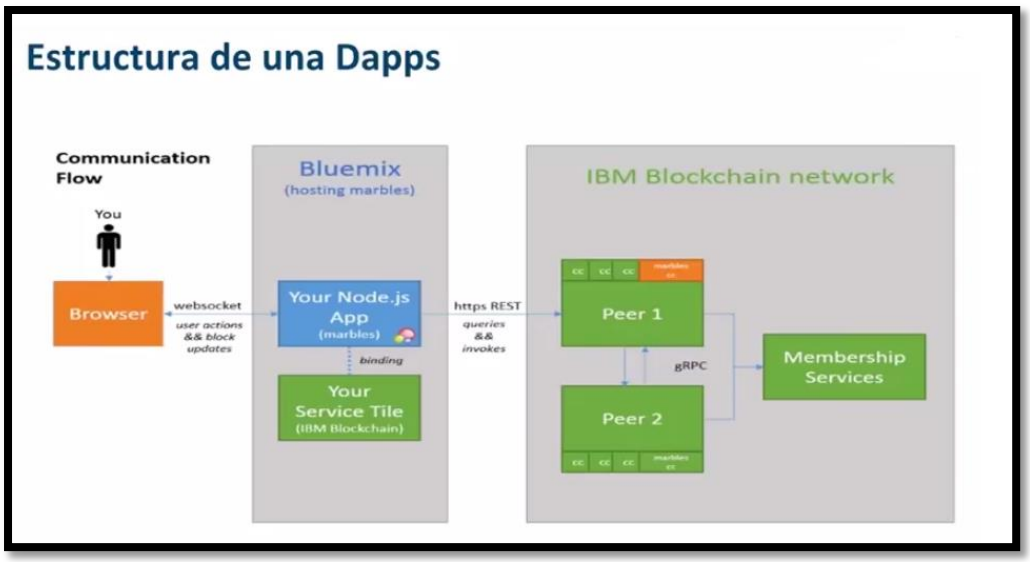


Ilustración 15 Estructura de una DAAP (Hyperledger , 2016)

Las DAAP son Decentralized Applications, las cuales funcionan sobre la Blockchain de cada proyecto específico.

Los casos de uso más conocidos entorno a HyperLedger:

Ilustración 16 Casos de Uso Blockchain (Medina, 2018)

WEIFUND. – Blockchain dedicada a la exclusivamente al crowdfunding, financiamiento para nuevos proyectos ya sean basados en Blockchain o para lanzamiento de ICO's. http://weifund.io/

SKUCHAIN. – Blockchain dedicada a la implementación de soluciones ERP para control de inventarios y contabilidad. http://www.skuchain.com/

4G CAPITAL. – Blockchain dirigida al mantenimiento y mejora de África y sus sectores necesitados, incluyendo a personas que no tienen los recursos ni financieros, ni de infraestructura con el fin de ingresarlos al sistema monetario soportando micro pagos. http://www.4g-capital.com/

VEVUE. – Blockchain dirigida a la creación de videos bajo pedido y compartirlos en la red y se reconoce sus valores por medio de pagos mediante Bitcoin. https://www.vevue.com/

LAZOOZ. – Es una aplicación basada en Blockchain, que pretende ser la competencia de UBER, ofreciendo costos que oscilan entre el 10% del valor de la competencia al igual que ARCADE CITY, pretenden destronar a la mundial UBER. http://lazooz.org/

Imaginar que se puede entre algunos inversores recolectar una cantidad y se compra un TAXI INTELIGENTE, y este presta los servicios a sus clientes, y por medio de contratos inteligentes, empieza a cobrar por el servicio cada segundo desde que el usuario se ha subido hasta que termina su viaje, de que este taxi se está quedando sin combustible y como tiene los recursos automáticamente se acerca hacia la estación de combustible más cercano y llena su tanque, luego paga con sus coins que posee de los pasajeros que ha llevado a su destino, al finalizar el día, reporta los ingresos, menos los egresos y realiza los pagos de impuestos al estado y al terminar los cálculos, reparte las utilidades diarias a la cuenta de cada uno de los inversores. –En teoría– ya se puede hacer el contrato inteligente, sólo falta la tecnología física (el vehículo inteligente y la cadena de bloques que las conecte).

2.4. Contratos Inteligentes

Para hablar técnicamente de los contratos inteligentes tenemos unos pre-requisitos que hacen falta en el conocimiento general de las personas que desean inmiscuirse en este mundo.

Se necesita conocimientos previos de qué son, y cómo funcionan las criptomonedas, si se desea programar contratos inteligentes por medio de Solidity, se necesita tener un conocimiento previo de programación, de preferencia conocer HTML, CSS y JavaScript, además de cómo funcionan los lenguajes de programación.

Los contratos son muy fáciles de escribir y ejecutar, pero son muy difíciles de escribir de forma segura, para asegurar los contratos inteligentes se deben aplicar normas y procedimientos y se recomienda utilizar los conceptos publicados en la web de (OpenZeppelin, 2018) openzeppeling.org para poder corroborar la seguridad de los mismos a manera de realizar auditorías de contratos inteligentes.

2.4.1. IPFS

Sistema de Archivos Intra Planetarios ‘Intra Planetary File System’, a pesar de tener un nombre elegante como dice (Simply Explained, 2018) en su presentación online, donde presenta una solución al problema de la centralización del internet, en la cual acota que toda la multimedia que se encuentra en la red está centralizada, es decir, si un servidor, por ejemplo de Google, Wikipedia, o YouTube se cae, pues el contenido que en el existía queda inaccesible, además, este sistema centralizado permite la censura hacia cierto contenido,

privando a naciones enteras su acceso, el sistema de archivos interplanetario propone un internet 100% P2P peer to peer, con el objetivo de que si algún archivo que existía en la red, alguien más lo pudo haber descargado antes, y solucionando el problema de la adulteración de archivos, utilizando una funciona hash de comprobación para que sea el mismo archivo el que se busca, sin cambios. También en base al sistema de archivos por función hash, se puede ubicar a los archivos y eliminar duplicados que no sean necesarios haciendo más eficiente la red.

Este sistema tiene una semejanza con una cadena de bloques en la manera en que almacena los archivos, almacenar archivos en una blockchain puede resultar sumamente costoso, pero es lo que IPFS pretende hacer, ha creado una moneda virtual llamada FILECOIN 'https://filecoin.io' la que se utiliza para compensar a usuarios que brindan su alojamiento para que los archivos siempre estén disponibles para los usuarios finales y que se encuentra listada en Coinmarketcap 'https://coinmarketcap.com/currencies/filecoin/'.

Ilustración 17 Filecoin en Coinmarketcap (coinmarketcap.com, 2018)

Casos de uso reales se dieron cuando el gobierno turco bloqueó Wikipedia y la levantaron en el sistema IPFS dando acceso sin poder ser bloqueados porque no tenían un servidor central, en cambio se levantó en el sistema P2P, además existe una página DTUBE 'https://d.tube/' en la cual se puede subir contenido y compartirlo apoyando a la red.

IPFS es un proyecto muy ambicioso que en el futuro desearía operar, como su nombre lo indica, entre planetas.

2.4.2. Swarm

Según la publicación de (Tron, swarm-gateways.net, 2018) Swarm funciona de manera similar a IPFS pero para uso exclusivo de la red Ethereum, es una plataforma de almacenamiento distribuida y un servicio de distribución de contenido utilizado por Ethereum web3. El objetivo principal es proporcionar un almacén descentralizado para datos y códigos Dapp, cadenas y estados de los bloques.

Swarm también brinda servicios de capa base para web3, como mensajes de nodo a nodo, transmisión de medios, bases de datos descentralizados y la infraestructura de un canal de estado para economías descentralizadas.

Las características principales que ofrece Swarm.

- Tolerante a fallos, posee almacenamiento redundante que aseguran la disponibilidad de los datos, ante abandonos de datos o perdida de nodos.
- Resistencia a la censura, los sitios creados no se pueden eliminar, los datos se almacenan en toda la red.
- Resistente a DDoS, es comprobado que la red P2P es más resistente a ataques de denegación de servicio que cualquier otra red centralizada.
- Cero tiempos de inactividad, la redundancia garantiza la entrega de los datos incluso cuando los nodos se hayan desconectado.
- Auto-sostenible, el sistema de incentivos incorporados garantiza la viabilidad económica de la red.

Swarm se puede instalar tanto en Linux como dentro de un contenedor Docker.

2.4.3. Web3.js

Para comprender que es y cómo funciona Web3.js nos referiremos al sitio web de (Lara, 2017) donde extraeremos un resumen del funcionamiento.

Web3.js es una API escrita en JavaScript compatible con Ethereum que implementa especificaciones del protocolo RPC en formato JSON.

Para que esta aplicación funcione en Ethereum se puede usar el objeto ‘web3’ proporcionado por la biblioteca web3.js

La librería se comunica con un nodo local a través de llamadas RPC.

Web3.js funciona cualquier nodo Ethereum que expone una capa RPC.

2.4.4. Token ERC20

Según (CRIPTOTENDENCIA, 2017) El estándar ERC20 es un estándar para que todos los Tokens que operen en este sistema sean fácilmente intercambiables. La utilización de este estándar es lo que permite a los programadores un cierto conjunto de funciones para que el proyecto realizado se pueda integrar de manera automática a la red Ethereum y no tenga malfuncionamiento.

Los Tokens en cada proyecto representan valor que puede ser intercambiado entre los usuarios directamente y afianzados por la verificación de transacciones por medio de la minería.

En resumen un Token es un contrato inteligente conformado por codificación estandarizada que funciona sobre la red Ethereum.

2.4.5. Solidity

Solidity es un lenguaje de programación de alto nivel '***Turing complete***', orientado a objetos utilizado para el desarrollo de contratos inteligentes en la red Ethereum 'en lugar de lenguajes de bajo nivel como 0 y 1 para que sean más comprensibles por el ser humano', el mismo que su base principal es la máquina virtual de Ethereum 'EVM Ethereum Virtual Machine' que se encuentra instalada en todos los nodos que integran la red.

Solidity cuenta con la comunidad más grande de desarrolladores adeptos a JavaScript.

En Solidity los 'contratos' son similares a las 'clases' en los lenguajes de programación orientada a objetos.

2.4.6. Metamax

Es una extensión de navegador especialmente utilizado en Google Chrome y Firefox, la cual permite crear cuentas y administrarlas, tanto para la red Ethereum o para la TestNet y que es

un complemento para el desarrollo de los contratos inteligentes que corren en un ambiente de sandbox totalmente aislados de los datos externos.

2.4.7. Ethereum Virtual Machine

La función de la EVM es compilar el código escrito en Solidity para hacer funcionar los contratos inteligentes, la EVM se encuentra totalmente aislada, lo que significa que se pueden realizar muchos tipos de pruebas o test antes de la publicación final en la red oficial, pero es más difícil de asegurar lo que la puede volver más vulnerable porque no se puede cuidar

2.5. Referencias a otros trabajos de investigación

Dentro de la biblioteca de la UNIR podemos encontrar trabajos relacionados con Bitcoin y criptomonedas, sin entrar en el contexto de la metodología estudiada en el presente trabajo que son los contratos inteligentes.

Como nos comenta (Ornubia Diaz, 2017) En su TFM de UNIR, “El Bitcoin, la nueva moneda virtual y su Ordenamiento jurídico”, que realiza un estudio de la Criptomoneda más popular, vista desde el enfoque del Derecho, haciendo una comparativa frente al dinero electrónico.

En este trabajo estudia las regulaciones, las diferencias entre dinero virtual y dinero electrónico, ventajas e inconvenientes del Bitcoin, la posible regulación española del Bitcoin, la regulación del Bitcoin en la Unión Europea, habla a demás sobre la legalidad o ilegalidad del Bitcoin a nivel internacional, para lo cual concluye:

1. La característica fundamental entre la moneda convencional y el Bitcoin es la descentralización e independencia de órganos o gobiernos encargados de su regulación.
2. La creciente tendencia de manera exponencial de la adopción de criptomonedas y su uso con respecto a los Bitcoins en circulación.
3. El valor del Bitcoin se estabilizará luego de la ruptura de su burbuja para tener una estabilización en sus precios.

4. La fácil conversión de monedas virtuales a dinero de circulación real hace que se pueda utilizar para actos fraudulentos, siendo este uno de los puntos que más preocupan a las autoridades al momento de afrontar la regulación.

5. Al ser los usuarios más activos y con más adopción requieren de una pronta regulación, incluso para proteger sus intereses ante una virtual falla del sistema.

6. Los problemas futuros a corto plazo de Bitcoin, según los gobiernos es presentar una utilidad práctica, ya que al momento solo se utiliza para pocos comercios y basados mayoritariamente en la especulación.

7. Los gobiernos se están preparando para la amenaza del Bitcoin y será brevemente condicionado para que tenga una aceptación jurídica.

8. La Unión Europea no considera aun relevante para que sea objeto de regulación y ésta surgirá de los informes regulatorios de la European Banking Authority.

9. España entorno a este caso, sigue a la regulación Europea, para ejercer una respuesta conjunta con el objetivo de evitar posibles discrepancias regulatorias sobre los usuarios que los poseen.

En el Trabajo de Fin de Master de UNIR (Yuste Gonzalez, 2015), titulado “Deep web y monedas virtuales: entorno privilegiado para las organizaciones terroristas” podemos resaltar lo siguiente:

1. Expone la idea de que Bitcoin es utilizado por el terrorismo para realizar las transacciones de manera anónima, básicamente por la facilidad que se presentaba para comprar objetos controlados o prohibidos por la ley dentro del sitio web Silk Road y demás mercados negros dentro de la Deep web como armas, explosivos, estupefacientes, servicios de hackers, servicios de sicarios, entre otros.

2. Con el auge de las criptomonedas se ha podido crear un nuevo servicio ‘CAAS crime as a service’ y ser pagado con Bitcoins ya que utilizan canales encriptados de comunicación y es casi imposible dar con los maleantes.

3. El uso de la ‘Dark Wallet’ mezcla los Bitcoins provenientes de distintas cuentas sin que sea posible al final determinar el origen de los ingresos, además los terroristas se pueden aprovechar de los negocios que ya aceptan la criptomoneda para obtener fondos y blanquear dinero no controlado.

4. La combinación de la Deep web y el Bitcoin es una ecuación difícil de controlar por las autoridades, en principio porque aún no se crean las leyes suficientes para dicho control, y es un hecho que se debe realizar a nivel mundial.

En el TFM de UNIR de (Ramírez Gracia, 2018) Hacia un tratamiento uniforme del Bitcoin desde una perspectiva fiscal global, tiene un enfoque legal analizando las criptomonedas y Bitcoin resaltando lo siguiente:

1. La economía digital ha dado un salto muy grande y veloz que a todos está tomando por sorpresa y es poco lo que se puede hacer al tratar en intentos desesperados por legalizar y controlar el sistema disruptivo que ha ingresado con las criptomonedas a nivel mundial.

2. Los países no se encuentran capacitados para integrar los Bitcoins dentro de la economía como títulos-valores, bienes muebles o como dinero, todo dependerá del régimen tributario aplicable a los mismos dentro de una determinada jurisdicción.

3. EL Bitcoin es utilizado como medio de pago alternativo al dinero, según expertos, se está al momento utilizando como instrumento de especulación, más que como medio de pago.

4. Estamos ante un fenómeno global sin precedentes que no solo se limita a las directrices de EEUU y la Unión Europea, no han tenido conceso a nivel transaccional ni al tratamiento fiscal.

5. Dado que por su naturaleza el Bitcoin no tiene fronteras y supera cualquier barrera física y legal, se debe tener el apoyo mancomunado de los demás países como China, Rusia, India y Brasil, a efectos de lograr acuerdos multilaterales para buscar un bloque regulador común que conduzcan a una estabilidad y normalización de mercado.

6. En España la Comisión Nacional de Valores advirtió recientemente que las inversiones en monedas virtuales son menos garantistas que los valores tradicionales, por lo que deberían cumplir con las protecciones que el mercado de valores exige.

7. Bitcoin se encuentra en su fase inicial del proceso de expansión y su alcance es imposible de determinar al momento. Para todos los servicios de recaudación de impuestos a nivel mundial les es muy difícil o casi imposible poder controlar estos movimientos.

8. No hay que alarmarse si por medio de esta tecnología dejaran de existir los bancos, cambios de moneda, limitar la economía de los Estados, resultaría aconsejable que los líderes mundiales consideren una actualización a los actuales sistemas y cambien su punto de vista.

En otros estudios a nivel mundial podemos obtener los siguientes conceptos más adaptados al estudio presente trabajo.

En la tesis de (Wall & Malm, 2016) de la universidad de Lund, Suecia, con el tema: "Usando la Tecnología Blockchain y los Contratos Inteligentes para crear un Depósito de valores distribuidos" '*Using Blockchain Technology and Smart Contracts to Create a Distributed Securities Depository'* podemos destacar lo siguiente:

1. En el mercado de valores, los depósitos de valores representan la capa más importante de la que dependen todos los otros sistemas. En el transcurso de varias generaciones, esta capa ha estado sujeta a un proceso de centralización cada vez mayor.

2. Una descentralización de la capa de depósito de la infraestructura de valores es una tarea extensa que requiere una amplia colaboración de la industria y debe implementarse gradualmente. Este cambio constituye un cambio fundamental en las finanzas con un impacto potencialmente revolucionario.

3. El desarrollo de un sistema distribuido interoperable de registros de código abierto que se encuentre coordinado y hecho para seguir estándares mutuos, podría abrir oportunidades para que todos los participantes en el mercado imaginen, innoven y construyan, no solo instrumentos financieros nuevos, sino sistemas totalmente nuevos que podrían comunicarse e interactuar directamente con el sistema financiero, o incluso reinventarlo por completo. Ese es el premio, y todo este viaje es posible gracias a la invención de Nakamoto y la cadena de bloques.

En el trabajo de investigación de (Bergquist, 2017) presentado en la Universidad de Uppsala, en Suecia, titulada: "Tecnología Blockchain y Contratos Inteligentes – Herramientas para

preservar la privacidad" '*Blockchain Technology and Smart Contracts - Privacy-preserving Tools'* podemos nombrar las siguientes ideas principales:

1. En esta tesis, se creó una aplicación de prueba de concepto para funcionar como un plan de medicación electrónica de forma completamente descentralizada. Para lograr una red peer-to-peer lo suficientemente segura como para almacenar información personal, se propuso un sistema de contratos inteligentes desarrollado en el lenguaje de programación Solidity en combinación con una arquitectura blockchain permitida y herramientas criptográficas comunes. El producto resultante se evaluó luego según los criterios de investigación de diseño establecidos y se encontró que cumplía con todos los requisitos necesarios. Aunque se cumplieron los criterios de evaluación, es importante tener en cuenta que no se pueden realizar reclamaciones sobre la seguridad fuera de PoC, de acuerdo con esta tesis. Hay formas en que se puede robar una clave privada, tenga en cuenta que la mayoría de los casos de fraude crediticio no solamente se realizan pirateando una computadora, sino simplemente restableciendo la contraseña de la cuenta de correo electrónico por teléfono.

2. En cuanto a las preguntas de investigación, todas han sido respondidas por la teoría resumida en la tesis, así como por la arquitectura de la aplicación y el razonamiento a su alrededor. Las preguntas de investigación se resumen brevemente aquí:

    a. Pregunta de investigación 1: ¿Cuáles son los requisitos para el almacenamiento de recetas, perfiles de pacientes, médicos y farmacias en una aplicación de blockchain para recetas?

    b. Pregunta de investigación 2: ¿Cómo debería ser la arquitectura de una aplicación blockchain para el intercambio de datos que preserva la privacidad entre partes conocidas pero no necesariamente confiables?

    c. Pregunta de investigación 3: ¿Cómo se puede construir una aplicación Blockchain para el manejo de recetas para garantizar que cada paciente tenga control de acceso a las recetas, que solo los médicos certificados puedan recetar medicamentos y que las farmacias que venden medicamentos puedan controlar las recetas?

3. La pregunta de investigación 1 se responde en la Sección 3.1 y las tablas en la misma. Además, se exponen los antecedentes de esos requisitos en el Capítulo 2. La pregunta de investigación 2 se responde en la Sección 3.2.1, donde se muestra el diseño de la implementación. Se explica una arquitectura posible de un plan de medicación electrónica, basada en la tecnología blockchain y un sistema de contratos inteligentes.

La pregunta de investigación 3 se responde a través del diseño del sistema de contratos inteligentes, así como de las opciones de discusión y diseño, hechas en la Sección 3.2.3 y en la Sección 4.2.1.

En la tesis de maestría de (Schüpfer, 2017), publicada en la Universidad de Suiza, con el título: "Diseño e Implementación de una aplicación de Contrato Inteligente" *'Design and Implementation of a Smart Contract Application'* podemos destacar los siguientes puntos:

1. La aplicación Android implementada satisface los requisitos funcionales y no funcionales establecidos al comienzo de esta tesis. Permite a dos partes celebrar contratos electrónicos de compra y alquiler válidos. Es completamente independiente de cualquier tercero de confianza y utiliza la cadena de bloques de Ethereum para almacenar los detalles del contrato, hacer cumplir las cláusulas contractuales y procesar los pagos en Ethereum. El vendedor puede especificar los detalles del contrato, incluido el precio, el depósito, la descripción textual y las imágenes, y desplegar el contrato en la cadena de bloques. Los detalles del contrato pueden intercambiarse escaneando el Código QR de un contrato o usando Wi-Fi directo. En este último caso, las partes pueden decidir de manera exclusiva qué información personal quiere compartir.

2. La aplicación firma contratos en el dispositivo Android local y protege los datos personales del usuario al almacenarlos solo en el almacenamiento interno y al encriptarlos cuando se envían a través de la red. La aplicación es altamente escalable y puede manejar una gran cantidad de contratos en paralelo. Detecta errores de red e impide que el usuario pierda dinero al almacenar contratos de manera preventiva cuando la red falla durante las transacciones de creación de contratos. Los contratos de la aplicación se pueden ampliar con nuevos atributos o funcionalidades con un esfuerzo moderado.

3. La aplicación es similar a la aplicación de intercambio descentralizada de Bogner et all. Ambas son aplicaciones móviles y dependen de un cliente externo de Ethereum para interactuar con contratos en el blockchain y ambas aplicaciones respaldan la conclusión de contratos de alquiler entre dos partes. La diferencia más importante es

que la aplicación para compartir descentralizada utiliza un contrato inteligente central para administrar los artículos de alquiler existentes y se centra más en el alquiler comercial de artículos de una tienda. La aplicación implementada en esta tesis se enfoca más en transacciones únicas directas entre dos partes y también admite contratos de compra.

4. Otra diferencia es que la aplicación de Bogner et all no admite el intercambio de información personal entre los participantes.

2.5.1. Conclusiones

- Luego de analizar el estado del arte, tanto a nivel local, trabajos en UNIR y en español, y también haber analizado a nivel mundial, trabajos en otros idiomas, no se han encontrado estudios que tengan una propuesta similar a la del presente trabajo de investigación, todos los trabajos anteriormente analizados tienen un contexto sobre cripto monedas en el aspecto legal, financiero, relacionados con la cyber delincuencia, en los de tipo analíticos, y en los trabajos de carácter práctico en los que hablan sobre los contratos inteligentes, la implantación de una aplicación sobre un sistema Android basado en contratos inteligentes, y por último encontramos la programación de contratos inteligentes en relación a objetos financieros, por consiguiente se justifica la realización del presente ensayo enfocado exclusivamente a la creación de una criptomoneda estable que pueda ser usada en contratos inteligentes sobre la red Ethereum.

3. Objetivos concretos y metodología de trabajo

3.1. Objetivo general

El objetivo general establecido para este TFM consiste en la creación de un TokenERC20 que se encuentre anclado a la cotización de una divisa nacional, como puede ser el Euro, Dólar, Libra Esterlina, Yen, Yuan, etc., Para el presente caso, utilizaremos la paridad del Dólar Americano frente a la criptomoneda host de la red Ethereum, denominada ETH, que al momento de escribir esta investigación, se encuentra en alrededor de $250 USD. Para qué se crea esta criptodivisa?, con el fin de poderla implementar en los contratos inteligentes, y que éstos puedan ofrecer un valor fijo, tanto al iniciar como al finalizar el pacto, puesto que en muchos casos, debido a la extrema volatilidad de las criptomonedas, los valores pactados en ellas, difieren mucho entre el inicio y el fin de los convenios, que afectan tanto a los intereses de los contratantes y de los contratados.

3.2. Objetivos específicos

Los lineamientos que se deberán realizar para poder cumplir con el objetivo general, serán aquellos que nos permitan analizar detalladamente el estado del arte para comprender en profundidad el problema, y de esta manera, a futuro, tener la capacidad para poder crear el TokenERC20 anclado a un valor fijo y poder utilizarlo en la creación de los contratos inteligentes con la cual se pueda superar el problema.

1. Realizar un estudio del arte sobre las criptomonedas y la blockchain en los cuales se basan los contratos inteligentes, identificando y comprendiendo sus características como su historia y contexto.

2. Identificar y estudiar las técnicas de creación de TokenERC20.

3. Identificar y estudiar el uso de los contratos inteligentes.

4. Identificar los actores que intervienen en el escenario de los contratos inteligentes.

5. Identificar y analizar las herramientas que se utilizan para la creación de Tokens y contratos inteligentes.

6. Implementar un entorno de prueba que nos permita simular un entorno real para comprobar la creación y funcionamiento de los contratos inteligentes sobre la TestNet de Ethereum.

7. Buscar la sistemática más eficaz y eficiente posible para la creación de los Tokens.

8. Identificar y analizar tendencias futuras en el campo de los contratos inteligentes y la transferencia segura de valores.

3.3. Metodología del trabajo

Se realiza un análisis de los programas y FrameWorks que conforman las técnicas que se utiliza para la creación de contratos inteligentes, entre los más utilizados se encuentran: Solidity (tema de estudio) y Truffle bajo la plataforma Ethereum, JavaScript se utiliza directamente en la plataforma Lisk y de manera indirecta por medio de web3.js en Ethereum, y en la actualidad se está implementando Cardano que ofrece la separación de los datos transaccionales de la información contractual.

La manera de probar el presente estudio es con la creación paso a paso de una criptomoneda o Token del tipo ERC20 que su valor se encuentre anclado a la de una moneda real específica, en este caso para la práctica se creará un Token anclado al valor del Dólar estadounidense, con el fin de que se puedan realizar los contratos inteligentes en la vida real y que exista a futuro una mayor adopción de los mismos por la gente común, ya que muchos de ellos en la actualidad se alejan de las criptomonedas por su alta volatilidad, citando un ejemplo.

Si se genera un contrato inteligente por un valor de $200 usd calculado en Ethereum ~ 0.60 ETH, y al finalizar el contrato en aproximadamente 2 semanas (por ejemplo) reciben su pago, los mismos 0.60 ETH equivalen a ~ $170 usd, la gente no adopta estos contratos por la volatilidad de precios. Y viceversa, si el precio sube, el perjudicado sería el contratante.

Al crear una moneda anclada, como es el caso de USDT (TETHER) o TUSD (TRUE USD), que en éstas blockchain aún no se pueden realizar contratos inteligentes, a pesar de que se encuentra listado en su ROADMAP, se pretende utilizar un TOKEN ERC20 llamado **E-USA** (E viene de Ethereum), que represente el valor de 1 dólar americano, a futuro, una propuesta sería la aplicación de una metodología para la creación de una moneda virtual en formato ERC20 para todas las denominaciones de las divisas de cada país, como lo represento en la siguiente tabla.

Tabla 6 Tokens Anclados a Moneda Nacional

SIMBOLO	EQUIVALENCIA
E-USA	1 DÓLAR AMERICANO
E-EUR	1 EURO
E-CHN	1 YUAN
E-JPN	1 YEN
E-GBR	1 LIBRA
E-ARG	1 PESO ARGENTINO
E-COL	1 PESO COLOMBIANO
E-MEX	1 PESO MEXICANO
E-BRA	1 REAL

E-AUS	1 DÓLAR AUSTRALIANO
E-CAN	1 DÓLAR CANADIENSE
E-KOR	1 WON

En el desarrollo se realizará los ejemplos de la metodología para la creación de la moneda E-USA que es equivalente a un dólar americano, y puede ser utilizada de contratos inteligentes ya que es un Token ERC20 y se puede manejar dentro de toda la plataforma Ethereum.

Estándares de ERC20

Los campos que obligatoriamente debe contener un SmartContract de creación de Token, para que cumpla con las normas mínimas de seguridad son los siguientes:

function totalSupply()
function balanceOf(address owner)
function transfer(address to, uint256 value)
function approve(address spender, uint256 value)
function allowance(address owner, address spender)
function transferFrom(address from, address to, uint256 value)

Luego de ejecutar y evaluar los métodos de la creación y las respectivas pruebas del Token, una propuesta a futuro puede ser la creación de la moneda, lanzando una ICO o solicitar el financiamiento de posibles inversionistas que se pueda plantear la idea a través de CrowdFounding, crear un Fideicomiso, etc.

En este momento nos vamos a centrar en el desarrollo y creación del Token que funcionará como una criptomoneda basado en la sistemática básica para crear correctamente un contrato inteligente utilizando Solidity y Remix, y que a la vez, esta moneda nueva y creada, soporta toda la estandarización de los contratos estableciendo el Token "**E-USA**" (**Ethereum USA**) como medio de pago y recompensa, utilizando los convenios o estándares para la creación de un Token ERC20.

Solidity utiliza la notación de JavaScript por lo que la forma de escritura y de insertar comentarios es similar.

4. Desarrollo específico de la contribución

Según la palabra desde el aspecto legal en su libro “Smart Contracts: Análisis Jurídico” (Tur Faúndez, 2018) expone de forma detallada para el completo conocimiento de los actores externos involucrados en derecho, sin experiencia informática, que desean conocer como inmiscuirse en la operación de los contratos inteligentes, ante la cual existe una intervención de Nick Zsabo ‘posible Satoshi Nakamoto’ en la que se pronuncia que con la creación de los contratos inteligentes se obviará el trabajo de los notarios y abogados, por lo que el autor Tur Faúndez, no está de acuerdo con esta aseveración, además expone que un contrato está sujeto a la ley y jurisdicción de cada país en el mundo, por lo que si solo se utilizan contratos inteligentes basados en SmartContracts dentro de las blockchain a futuro se verán fácilmente rechazados y podrían influir en pérdidas millonarias porque no han sido previamente registrados ‘legalmente’ bajo la ley de cada país, y estos podrían quedar solamente como acuerdos sin valor legal, por lo que en su libro propone la creación de los Contratos Legales Inteligentes ‘Smart Legal Contracts’.

A continuación podremos observar sobre el ejemplo de su libro (Tur Faúndez, 2018) una ilustración meticulosa del contenido de un Smart Legal Contract ‘Contratos Legal Inteligente’. Sobre el Arrendamiento de un vehículo inteligente, que cuenta con conexiones de tipo IoT, para lo cual nos ubicaremos en la perspectiva adecuada para poder entender la temática del evento.

Se procede a la especificación completa del contrato inteligente, y luego se explica detalladamente línea por línea cual es la acción que se realiza.

Antecedentes:
Contrato Arrendamiento de Vehículo

Parámetros:
Nos centramos en una empresa que posee una flota de vehículos autónomos Inteligentes, cada uno cuenta con conexión de internet, bloqueo y desbloqueo de motor y puertas mediante claves criptográficas, cuenta con sistema de rastreo satelital con registro en sistema blockchain, cuenta con conexión directa con las autoridades (policía o bomberos) por medio de sistema blockchain, cámaras a bordo, etc., la empresa posee una aplicación móvil que cuenta con una interfaz gráfica para que el usuario pueda ingresar los datos solicitados, como

su nombre, cuenta de criptomonedas, valor a pagar, aceptación SI o NO, un generador y scanner de códigosQR, un altavoz que pueda comunicar mensajes de voz al usuario.

Se utiliza la metodología de creación de contratos inteligentes con la herramienta Remix utilizando el leguaje Solidity, de la siguiente manera.

Se ingresa a la página web de remix “https://remix.ethereum.org/”

4.1. Piloto Experimental

4.1.1. Código:

```
pragma solidity ^0.4.19;

contract arrendamientoVehiculo {

  address public propietario;
  address public arrendatario;
  uint public plazo;
  uint public precioUnitario;
  uint public fechaCreacionSmartContract;
  string public IDvehiculo;
  string public matricula;

  enum Estado {Aceptado, Disponible}
  Estado public estado;

  function arrendamientoVehiculo(uint _precioUnitario, uint _IDvehiculo, uint _matricula) {
      propietario = msg.sender;
      arrendatario = 0x0;
      plazo = now;
      fechaCreacionSmartContract = now;
      precioUnitario = _precioUnitario;
      IDvehiculo = _IDvehiculo;
      matricula = _matricula;

  }

  modifier soloPropietario() {
      if(msg.sender != propietario) throw;
      _;
  }

  modifier soloArrendatario() {
      if(msg.sender != arrendatario) throw;
      _;
  }

  modifier queEstado(Estado _estado) {
      if(estado != _estado) throw;
      _;
  }

  function dimeID() constant returns (string) {
      return IDvehiculo;
  }

   function dimeMatricula() constant returns (string) {
      return matricula;
  }

```

```
50    function dimeArrendatario() constant returns ( address ) {
51        return arrendatario;
52    }
53
54    function dimePrecioUnitario() constant returns (uint) {
55        return precioUnitario;
56    }
57
58    function dimeFechaCreacionSmartContract() constant returns (uint) {
59      return fechaCreacionSmartContract;
60    }
61
62    function dimeCuentaSmartContract() constant returns (address) {
63        return this;
64    }
65
66    function dimeEstado() constant returns (Estado) {
67        return estado;
68    }
69
70    event contratoAceptado();
71    event vehiculoDisponible();
72
73    function alquilar() soloArrendatario payable returns (bool){
74    queEstado(Estado.Aceptado);
75    require(msg.sender != propietario);
76    {
77        if(arrendatario != 0x0) return false;
78        else{
79            if((msg.value/ 1) / precioUnitario < 1) return false;
80            else{
81                plazo = now + (1 minutes * (msg.value/precioUnitario));
82                fechaCreacionContrato = now;
83                arrendatario = msg.sender;
84                Estado = Estado.Aceptado;
85            }
86        }
87    }
88
89    function enUso(bool _quiereAlquilar) payable returns(bool){
90        if(plazo > now) return true;
91        else {
92            arrendatario = 0x0;
93            Estado = Estado.Disponible;
94            vehiculoDisponible();
95            if(quiereAlquilar == true) alquilar();
96        }
97    }
98
99    function cobrar() soloPropietario {
100       propietario.send(this.balance);
101   }
102
103   function destruir()
104   queEstado(Estado.Disponible) { if (msg.sender==propietario) selfdestruct(propietario);
105   }
106
107 }
```

4.2. Explicación detallada del código:

Según (Tur Faúndez, 2018) la metodología para la creación de los contratos inteligentes, deben cumplirse los siguientes parámetros básicos para que la sintaxis del código pueda ser

ejecutada y comprendida por el ordenador y se puedan ejecutar de manera correcta los comandos enviados a consola, se explica línea por línea el código.

1. Se ingresa el comando: “pragma solidity ^0.4.13;” el mismo que nos indica la versión en la que se encuentra codificado el presente contrato inteligente, a la fecha de la realización del trabajo, la última versión es la 0.4.25 y no debe olvidarse de colocar el punto y coma al final del comando al estilo de la programación JavaScript ya que nos mostrará un error en la ventana de compilación.
2. Espacio en blanco, que se utiliza para separar elementos del contrato y para brindar una mejor lectura del código.
3. Utilizamos la palabra reservada “contract” que es con la cual daremos el nombre a nuestro Smart Contract, para el presente caso es “arrendamientoVehiculo” el mismo que será escrito bajo la notación técnica de la ‘escritura camello’, conocida de esa manera por la forma de escritura sin separaciones y sin tildes, y al final se coloca la llave de apertura “{” que en donde contendrá todo el contenido del contrato, hasta la línea 107 donde se hará el cierre de la llave “**}**”, con la finalización del Smart Contract **arrendamientoVehiculo**.
4. Espacio en blanco.
5. Definimos la primera variable en la memoria del programa, si añadimos la palabra reservada “public” esta variable se podrá ver desde otros Smart Contracts y puede ser llamada desde cualquiera, existe la posibilidad de poner public, private, o sin la notación es similar a colocar private. La primera variable es de tipo address (dirección de criptomoneda), la variable será: **address propietario public;**, corresponderá a la cuenta de la persona dueña del vehículo.
6. En la segunda variable se define la cuenta de la persona que va a rentar el vehículo con la línea: **address arrendatario public;**. En ambos casos se coloca la palabra reservada public para que se puedan conectar las cuentas con otros contratos inteligentes que se puedan enlazar, como páginas de calificación de usuarios como Airbnb o Uber.
7. Se define la variable **plazo** que se determinara en el futuro dependiendo del valor que haya ingresado el arrendatario, ésta variable es de tipo **uint** ‘unsigned integer’ entero sin signo, es decir que es un número entero que no puede tener valores negativos.
8. Se define la variable **precioUnitario**, que será un valor pactado en criptomoneda por cada minuto de arriendo del vehículo y que solo podrá ser definido por el **propietario**.
9. Se define la variable correspondiente a la fecha de creación del contrato, denominada **fechaCreacionSmartContract;** ésta se definirá automáticamente en el transcurso del programa cuando el arrendatario pague la cantidad correspondiente y se hará

referencia al día, hora, minuto y segundo en que la transacción sea introducida en un bloque valido de la cadena de bloques.

10. Se define la variable **IDvehiculo** con la cual se podrá identificar el vehículo que se encuentre disponible en el tiempo y el lugar para ser arrendado, ésta variable se consigue al utilizar un sistema de registro electrónico de cada vehículo, esta variable es de tipo **string** y puede alojar una cadena de texto.
11. Se define la variable **matricula**, también con el objetivo de la ubicación del vehículo y su disponibilidad, es de similar tratamiento que la anterior variable.
12. Espacio en blanco.
13. Con la palabra reservada **enum** se pueden crear variables que no están predeterminadas en Solidity, se crea la variable Estado y entre llaves "{ }" definimos los estados de ésta variable que son **Aceptado** y **Disponible**, la utilización de éstas variables de estado son muy importantes para definir el lapso temporal que en el contrato se deben ejecutar las funciones. Si no hubiera un estado que limite el tiempo, el contrato se podría realizar antes de lo pactado y el propietario podría cancelar el contrato sin que hubiese terminado el periodo por el cual fue pagado.
14. Luego de haber creado la variable *Estado*, se procede a denominar la variable con el nombre **estado** "tomar en cuenta la capitalización de las letras, solidity es sensible a mayúsculas y minúsculas".
15. Espacio en blanco.
16. En ésta línea empieza la creación de la función con el mismo nombre del Smart Contract, esto quiere decir que es un constructor y se ejecuta para atribuir los valores iniciales a cada una de las variables.

    Según la traducción de la guía (EthereumEng, 2016) de convención para escritura de contratos inteligentes en solidity adjunta en el Anexo 4, (Ethereum, 2017), procedemos a crear la función de la siguiente manera: **function arrendamientoVehiculo(uint _precioUnitario, uint _IDvehiculo, uint _matricula) {** donde se indica que los valores entre el paréntesis son los valores que se introducen en las funciones y sirven para determinar los parámetros del contrato. Son valores que se deben introducir al crear el contrato, en este caso se tratara de dos números enteros sin signo **uint** y una cadena de alfanumérica **string**.

    Los parámetros sirven para proporcionar al Smart contract el valor correspondiente a una variable, se identifican del mismo modo que ésta, pero con un guion bajo delante **_precioUnitario**, etc., de acuerdo a las convenciones de código anteriormente explicadas, al final de la línea existe la llave de apertura de la función "{" y que se cerrará en la línea 25.

17. La variable **propietario** es una **address**, una cuenta, que pertenece al propietario creador del contrato, se le atribuye por tanto el valor de **msg.sender** que en Solidity hace referencia a la cuenta que ha anclado el contrato a la cadena de bloques.
18. Definimos el valor de la variable **arrendatario** en 0x0 (según convención significa 0, cero, nada, ninguno) en solidity no existe el valor nulo (null), este valor se integrará al contrato cuando el arrendatario suscriba el contrato, entonces aparecerá ahí su cuenta.
19. Se define la variable **plazo** con el valor **now**.
20. Se define la variable **fechaCreacionSmartContract** con el valor **now**. El contrato inteligente interpreta la fecha y hora, y el plazo con la fecha en que se registre el contrato con la correspondiente aceptación de valores y el tiempo automáticamente empieza a correr.
21. Se asigna a la variable **precioUnitario** el parámetro **_precioUnitario**, con el fin de que este espacio en memoria será cargado al momento de la suscripción del contrato, de la misma forma con las líneas 22 y 23.
22. Se asigna a la variable **IDvehiculo** el parámetro **_IDvehiculo**.
23. Se asigna a la variable **matricula** el parámetro **_matricula**.
24. Espacio en blanco.
25. Cierre de llaves de la función constructor iniciado en la línea 16 "**}**".
26. Espacio en blanco.
27. Se insertan modificadores 'funciones especiales que proveen permisos o restricciones' que son los que dan los permisos para que no todos los participantes del Smart Contract puedan realizar modificaciones, hay un modificador para el propietario, otro para el arrendatario y un último para el estado del contrato de la siguiente manera.
    Se inserta la palabra reservada **modifier** y un nombre identificativo, en este caso **soloPropietario**, y se continúa con un paréntesis vacío para que se ejecute la función seguido de la apertura de llaves "**{**" donde estará el contenido de la función en la siguiente línea.
28. Se crea el contenido que es un 'si condicional' donde se ocupa la función if de la siguiente manera, if(msg.sender != propietario) **throw**; que significa si el msg.sender no es igual a la cuenta del creador, desechar;
29. _; en solidity el guion bajo significa **else** o entonces, lo que se traduce, entonces ejecútese.
30. Cierre de llaves "**}**"
31. Espacio en blanco.
32. Modifier del arrendatario, que si no es el arrendatario el que envía, se deseche, pero si es él que se ejecute.

33. Función si condicional.
34. Guion bajo en representación de else, _;
35. Cierre de llaves "**}**"
36. Espacio en blanco.
37. Modifier con relación al Estado, que para conocer el estado en el que se encuentra el vehículo, Disponible o Aceptado, **modifier queEstado(Estado _estado) {**
38. Función si condicional, **if(estado != _estado) throw;**
39. Guion bajo en representación de else, **_;**
40. Cierre de llaves "**}**"
41. Espacio en blanco.
42. De la línea **42** a la línea **68** se conforma el contrato de los Getters o informadores, que no realizan modificaciones pero que informan a través de la interfaz de usuario el valor de las variables solicitadas, para lo cual se ejecuta la palabra reservada **function** y se le asigna un valor con un nombre representativo que nos brinde una idea del valor que se desea obtener, seguido de un paréntesis vacío para que se ejecute la orden. Por ejemplo: **dimeID()**.

    A continuación se incorporan los términos **constant returns** que simulando una traducción sería algo como '**devuelve el valor constante**', y seguido del tipo de dato que se solicita (**string, address, uint,** etc.) y se finaliza la línea con una llave de apertura "**{**".
43. Retorna el valor de la variable, con la palabra reservada **return** y la variable, **return IDvehiculo;**.
44. Cierre de llaves "**}**".
45. Espacio en blanco.
46. Getter function **dimeMatricula() constant returns (string) {**
47. Retorno de valor **return matricula;**
48. Cierre de llaves "**}**".
49. Espacio en blanco.
50. Getter function **dimeArrendatario() constant returns (address) {**
51. Retorno de valor **return arrendatario;**
52. Cierre de llaves "**}**".
53. Espacio en blanco.
54. Getter function **dimePrecioUnitario() constant returns (uint) {**
55. Retorno de valor **return precioUnitario;**
56. Cierre de llaves "**}**".
57. Espacio en blanco.
58. Getter function **dimeFechaCreacionSmartContract() constant returns (uint) {**

59. Retorno de valor **return fechaCreacionSmartContract;**
60. Cierre de llaves "**}**".
61. Espacio en blanco.
62. Getter function **dimeCuentaSmartContract() constant returns (address) {**
63. Retorno de valor **return this;** aquí un cambio importante, está solicitando la cuenta del Smart Contract, entonces tiene que retornar ESTA cuenta determinada con la palabra reservada **this**.
64. Cierre de llaves "**}**".
65. Espacio en blanco.
66. Getter function **dimeEstado() constant returns (Estado) {** aquí otro cambio por el tipo de variable de estado que fue creada,
67. Retorno de valor **return estado;**
68. Cierre de llaves "**}**".
69. Espacio en blanco.
70. Se generan los eventos, un evento es la señal que facilita la comunicación, ya que pueden ser leídos directamente desde la interfaz del usuario y pueden ser escuchados por aplicaciones externas.

    Un evento es un disparador que hacer que las cosas sucedan, y que el programa se ponga en acción.

    En la línea 70 se ha generado un evento que corresponde con uno de los dos estados que puede tener nuestro Smart Contract.

    Los eventos posibles fueron definidos en la línea 13 como Estado.Aceptado y Estado.Disponible.

    event contratoAceptado(); indica que el contrato ha sido aceptado y está vigente.
71. event vehiculoDisponible(); indica que el contrato anterior ha finalizado y el vehículo se encuentra disponible para contratarlo.
72. Espacio en blanco.
73. En esta línea se consolida el pago, consentimiento y aceptación del arrendatario sobre la ejecución del contrato, creando la función determinante y que se va a denominar function quiereAlquilar() donde la opción a tomar como respuesta es SI o NO.

    En el caso de aceptar, nuevamente al usuario se le mostrará las opciones para alquilar, como son precio de costo en criptomonedas por minuto, si está conforme con el precio, puede nuevamente responder SI o NO.

    En el caso de responder SI, se inicia la transferencia de las criptomonedas desde la cuenta (address) del arrendatario, hacia la cuenta del Smart Contract.

    La función **alquilar() soloArrendatario** seguido de **payable returns (bool),** los operadores booleanos representan exclusivamente dos valores (si – no / verdadero -

falso), cuando se utiliza el término **payable** se le indica al SmartContract que para activar la función, necesariamente debe transferirse un valor para que el programa interprete como que se devuelve el valor **true** (verdadero).

Según los modificadores de la línea 37 y siguientes, sirven como activadores para que se dispare el contrato.

Se distingue que la función **alquilar(),** sólo estará operativa si se encuentra en **Estado.Disponible**, que es el estado por defecto cuando el vehículo no está alquilado.

74. Se dispone que el estado '**queEstado**' cambie de disponible '**Estado.Disponible**' a aceptado '**Estado.Aceptado**' con la ejecución del comando anterior.
75. La palabra reservada **require** y lo que se encuentra dentro del paréntesis es un requerimiento que se debe cumplir para que actué la función en este caso el requerimiento es: **(msg.sender != propietario)**. Para que la función se active, el programa verifica que, para rentar el vehículo, tiene que ser una cuenta diferente del creador del SmartContract.
76. Llave de apertura "**{**" que indica que todo lo que se encuentre dentro hasta la línea 87, son las operaciones de la función **alquilar()**.
77. Si el arrendatario **es diferente** (!=) de la **address** 0x0, desplegado en la línea 18, entonces que devuelva el valor **falso**.
78. **else** (traducido 'si no es así') y la apertura de llave '**{**' para crear una nueva función interna, si no se cumple la condición de la línea anterior.
79. Inicia con un **if** condicional anidado al **else** anterior, la palabra reservada '**msg.value**' se utiliza para designar el valor transferido por '**msg.sender**', en este caso el arrendatario.

    La expresión '**if((msg.value/ 1) / precioUnitario < 1) return false;**' se la interpreta de la siguiente manera: si el valor que envía el **arrendatario** dividido entre una unidad de criptomoneda, y eso dividido entre el precio unitario del minuto de arrendamiento del vehículo es menor que uno, retorna **falso**.
80. Nuevamente se utiliza **else** (traducido 'si no es así') y la apertura de llave '**{**' para crear una nueva función interna, si no se cumple la condición de la línea anterior. Que, si es menor a 1 criptomoneda que cuesta el minuto de alquiler del vehículo, retorna falso, es decir que si se hace el depósito de una cantidad mayor se ejecuta el contrato con las siguientes líneas de código.
81. Se activa la función **plazo = now + (1 minutes * (msg.value/precioUnitario));** que nos indica que si el '**msg.sender**' es decir, el **arrendatario** ha enviado '**msg.value**' una cantidad mayor al mínimo requerido, se active el **plazo** desde ahora donde se cuentan los minutos y segundos, hasta el valor que haya pagado el **arrendatario**.

82. Se asigna el valor definitivo que corresponde a los minutos y segundos a la del bloque que se valide en la blockchain y se confirme el pago.
83. Se asigna el nuevo valor a la variable **arrendatario = msg.sender**, la cuenta que envió las criptomonedas.
84. Y se cambia el estado a aceptado, **Estado = Estado.Aceptado;**
85. Cierre de la llave "**}**" de la línea 80.
86. Cierre de la llave "**}**" de la línea 78.
87. Cierre de la llave "**}**" de la línea 76.
88. Espacio en blanco.
89. La función **enUso** sirve para determinar si el vehículo se encuentra disponible, ya que al final de la función se puede observar que lleva anidada la function **alquilar();** lo que permite la interacción entre ambas.
    La function **enUso** requiere un parámetro **booleano** (**_quiereAlquilar**) con dos respuestas **true/false**, la función es **payable**, por lo que se activara solamente si el **msg.sender** 'arrendatario' envía fondos.
    Termina con una llave de apertura "**{**" y una de cierre en la línea 97.
90. Se establece una condición **if** que si el **plazo** es mayor que **now**, retorne **true**; verdadero, entonces se traduce como que si el plazo no ha concluido, el vehículo se encuentra en uso.
91. Se habilita un **else** para que si la condición anterior no se cumple, se abra una nueva llave "**{**" y se cumplan las funciones en ella escritas.
92. Si el plazo es **now** 'ahora' o si nadie ha arrendado el vehículo, o si el ultimo arrendatario ha concluido el periodo de arriendo, se le está indicando al programa que **arrendatario = 0x0;** el vehículo está disponible.
93. Entonces se le indica al programa que tiene que cambiar el Estado a disponible con la expresión **Estado** = **Estado.Disponible;**
94. Se solicita al programa que la función anuncie el evento de la línea 71, **vehiculoDisponible();** esto hará que se envié un mensaje de difusión anunciando su disponibilidad.
95. Cuando termine el uso del arrendatario actual, el SmartContract presentará una función si condicional, preguntando si desea continuar el arrendamiento, SI o NO, con la expresión **if(quiereAlquilar == true) alquilar();** la expresión de doble == significa que debe ser igual, la expresión de un solo =, es una asignación. Por lo tanto si el arrendatario actual acepta y remite los fondos, el contrato vuelve a ejecutar la función alquilar de la línea 87.
96. Cierre de la llave "**}**" de la línea 91.
97. Cierre de la llave "**}**" de la línea 89.

98. Espacio en blanco.
99. Inicio de la función cobrar o liquidar los valores del arrendamiento con la '**function cobrar() soloPropietario {**' el contrato inicia el envío de los valores recaudados a la cuenta de **quien creo** el contrato inteligente.
100. Se solicita al SmartContract que envíe los valores con la instrucción '**.send**' y que envié el total que hay en '**this.balance**', que se refiere al monto total de la cuenta del contrato inteligente.
101. Cierre de la llave "**}**" de la línea 99.
102. Espacio en blanco.
103. Inicio de la función destruir, una característica fundamental en la red Ethereum es la inmutabilidad, eso quiere decir que un SmartContract o programa podría estar siempre ocupando espacio en la cadena de bloques, por lo tanto, si se encuentra mal construido nunca podrá desaparecer, para eso, una de las **metodologías** que se debe tomar en cuenta es prever que, en las convenciones se estipula que debe existir una función **destruir** para que el contrato conste únicamente en los bloques en los que tuvo utilidad.

    Utilizando la expresión **function** y la descripción **destruir();** iniciamos el proceso de destrucción para finalizar el contrato.
104. Primeramente se verifica que el vehículo no esté en uso actualmente, con el modifier queEstado creado en la línea 37, solicitando o verificando que la variable estado debe estar en Disponible (**Estado.Disponible**) de lo contrario se puede terminar el contrato antes de que finalice el mismo.

    Se abre una llave de apertura "**{**" y se indica la función:

    '**if(msg.sender==propietario) selfdestruct(propietario);**' que significa, si la cuenta '**address'** pertenece al propietario, remítele todos los fondos en tu cuenta 'del SmartContract' destrúyete y desaparece.

    Cuando se ha ejecutado esta función, el SmartContract y su **cuenta** '**address'** pasan a ser definitivamente inaccesibles, pero el registro de su actividad quedará registrada siempre en la cadena de bloques donde existió.
105. Cierre de la llave "**}**" de la línea 104, termina función **destruir**.
106. Espacio en blanco.
107. Cierre de la llave "**}**" de la línea 3, cierra el contenido del contrato inteligente **arrendamientoVehiculo**.

Con la anterior explicación del código se hace referencia a la creación de los contratos inteligentes sobre la red Ethereum, siguiendo las convenciones estipuladas señaladas en el Anexo 4, al final del documento.

4.3. Identificación de requisitos

Para proceder a la realización del piloto se requiere la instalación del software necesario para que el IDE '*Integrated Development Environment*' de Remix de Solidity funcione de manera correcta, para lo cual utilizaremos un navegador o browser compatible, Chrome, Firefox, Opera, etc., para el presente trabajo se ha escogido Google Chrome, por su versatilidad y su compatibilidad con las extensiones necesarias requeridas.

Se procede a ingresar a la dirección "https://remix.ethereum.org/" y obtenemos la siguiente interfaz:

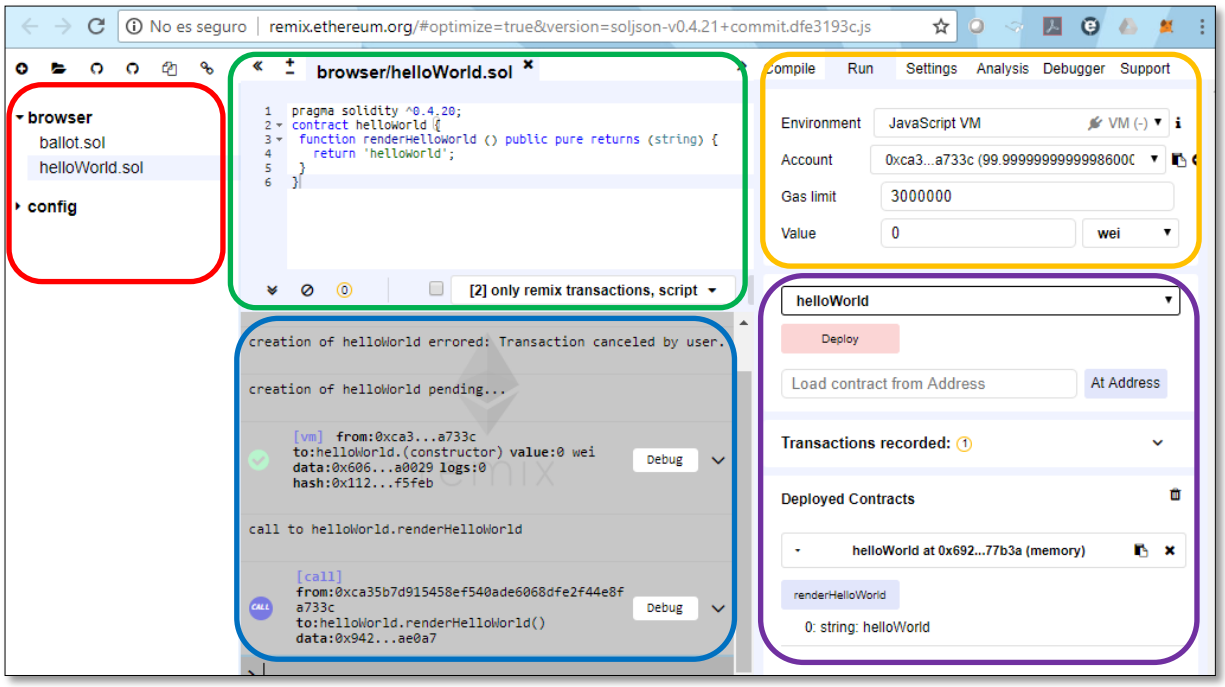


Ilustración 18 Remix - Solidity (ethereum/remix, 2018)

- Browser o navegador de documentos de extensión .sol
- Capa de editor de código.
- Área de compilador, ajustes, debugger, run o ejecución.
- Consola de control y logs.
- Área de Deployed Contracts, contenido del contrato

Para trabajar en la TestNet o en la Red Real de Ethereum se necesita un transportador que utiliza un medio de comunicación entre la EVM y la red Ethereum, ya que la EVM es completamente aislada de toda la red, y solo se puede comunicar mediante estas herramientas, en nuestro caso hemos utilizado la extensión de Chrome Metamax, que en el estado del arte en el marco teórico podemos encontrar más detalles.

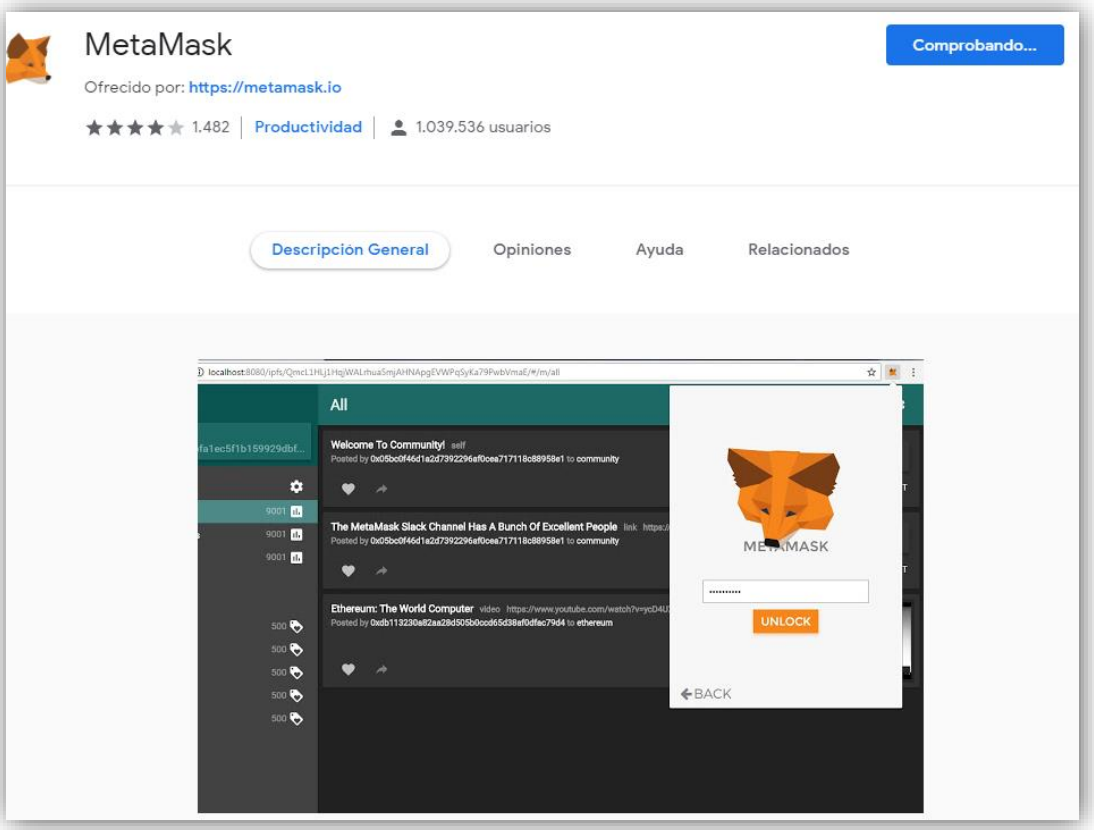


Ilustración 19 MetaMask (Chrome Webstore, 2018)

Una vez instalado nos colocamos en una de las redes que nos proporciona MetaMask, para nuestro caso elegimos Ropsten Test Network, con el fin de realizar todo en el ambiente de pruebas ya que en la Main Ethereum Network, el despliegue de contratos tiene un costo, en base a GAS como ya se ha explicado y se puede encontrar sus definiciones en el Anexo 1, Glosario de términos.

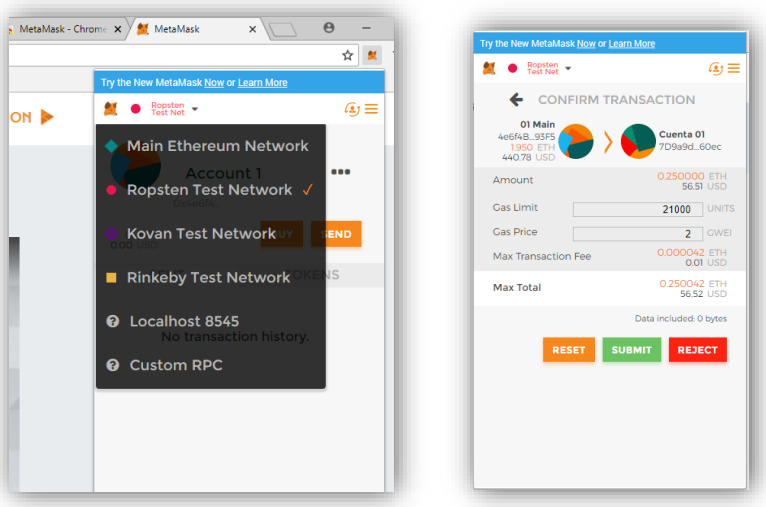


Ilustración 20 Selección de Red MetaMask (Chrome Webstore, 2018)

Posteriormente para realizar las pruebas cargamos la billetera '*Wallet*', la cuenta de TestNet desde las cuentas Faucet que nos permite obtener unidades de Ethereum de prueba para desarrollar los contratos sin tener que gastar valores reales, pero con todas sus funcionalidades, tenemos las 2 más conocidas, faucet.ropsten.be, faucet.metamask.io

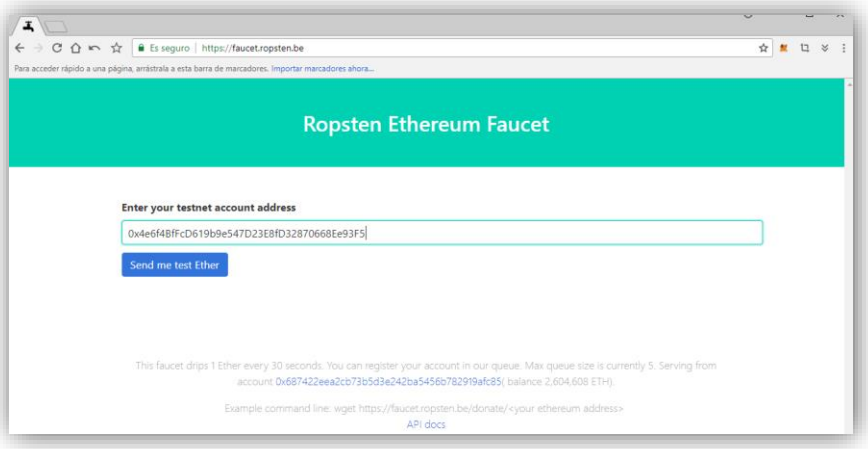


Ilustración 21 Faucet Ethereum Test (API docs, 2018)

Una vez cargado las wallet's en la red de prueba con ETHt (Ethereum Test) podemos proceder a hacer el despliegue de los contratos inteligentes.

Para el presente ejemplo creamos 4 cuentas

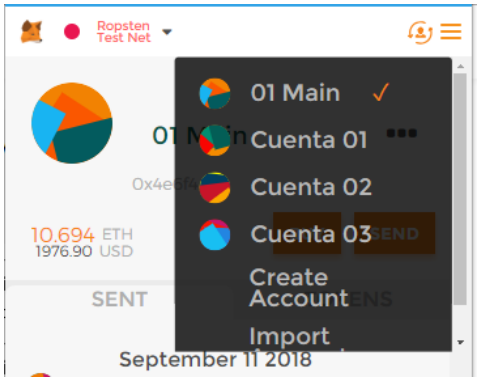


Ilustración 22 Cuentas de TestNet (Chrome Webstore, 2018)

01 Main (Principal y creadora del contrato y tenedor de Tokens)

Y tres cuentas de usuarios normales con las cuales se interactuará:

Cuenta 01 Empleado 01 Sueldo $ 1500
Cuenta 02 Empleado 02 Sueldo $ 1200
Cuenta 03 Empleado 03 Sueldo $ 1000
Cuenta externa de Proveedor pago $ 5000

Primero:

Ejecutamos el contrato inteligente, seleccionando desde MetaMax la dirección principal “01 MAIN” como se muestra en la ilustración 22.

En la sección de Deploy, encontramos 4 contratos:

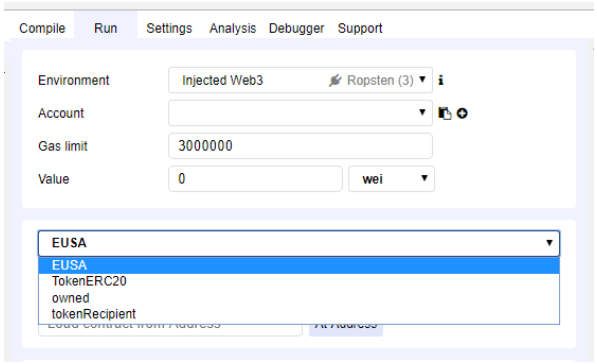


Ilustración 23 Solidity Deploy Contract (ethereum/remix, 2018)

Seleccionamos EUSA que es el contrato que creará los Tokens, para lo cual se proporciona la cantidad de las monedas circulantes, el nombre de la moneda, y el símbolo que utilizará el Token, en este caso: 1. 700.000, 2. Ethereum-USA, 3. E-USA. Y damos al botón ‘***transact***’.

Para lo cual se confirma la creación desde MetaMax en la creación del contrato inteligente confirmando sobre el botón '***submit***'.

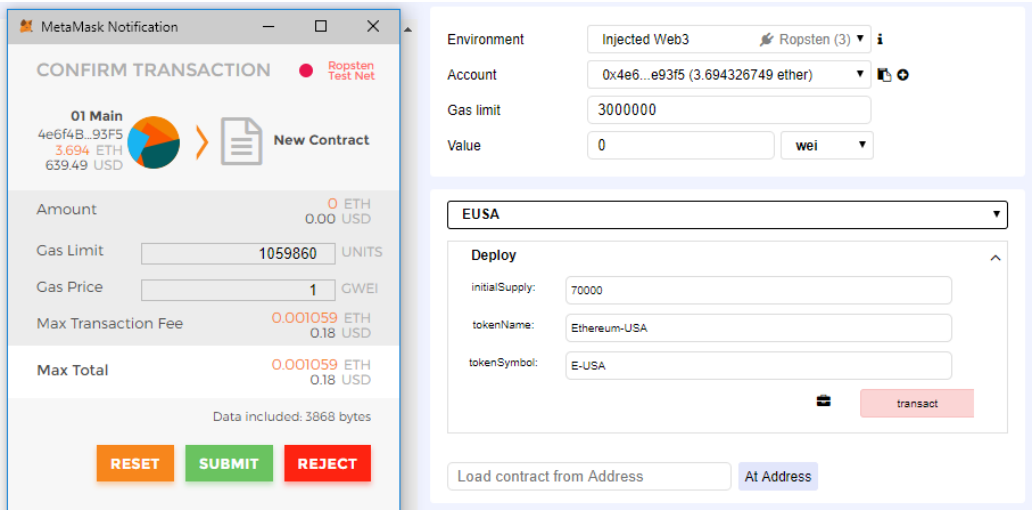


Ilustración 24 Creación de SmartContract (ethereum/remix, 2018)

Esta acción creara una difusión de la transacción a los mineros de la red, los cuales generaran un hash que queda registrado en la blockchain, en la cual constarán los valores antes mencionados y no podrán ser modificados jamás.

Se presenta en la consola de control y logs de la siguiente manera:

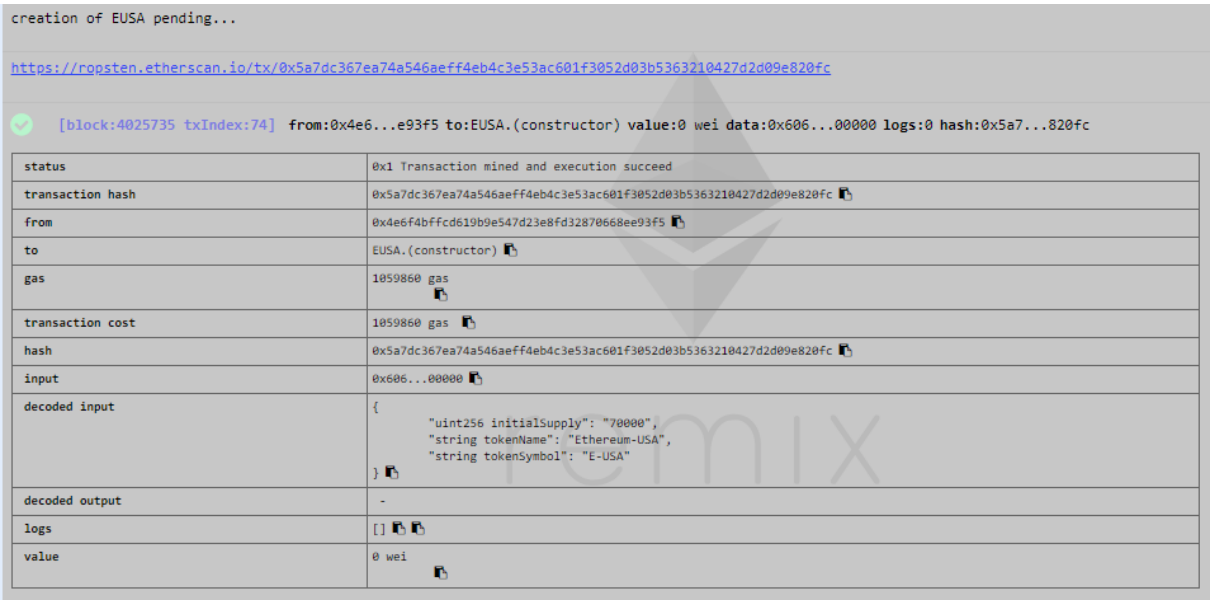


Ilustración 25 Transacción en Consola Solidity (ethereum/remix, 2018)

Y también se puede verificar dentro de la blockchain siguiendo el link:

https://ropsten.etherscan.io/tx/0x5a7dc367ea74a546aeff4eb4c3e53ac601f3052d03b5363210427d2d09e820fc

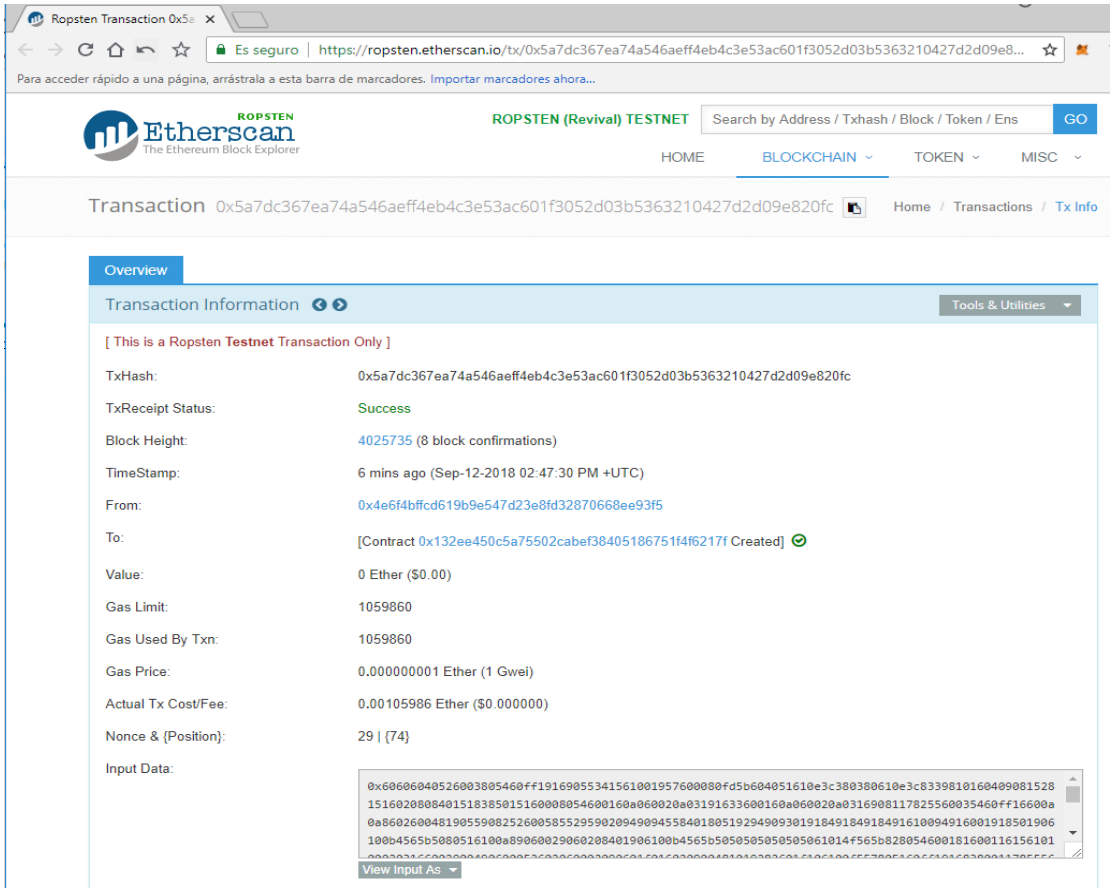


Ilustración 26 Transacción en Blockchain (Etherscan, 2018)

En esta transacción se ha creado el contrato con una dirección hexadecimal única para la localización dentro de la red, 0x132ee450c5a75502cabef38405186751f4f6217f, el mismo que contiene la información de la creación del contrato, direcciones de creación y destino de los participantes.

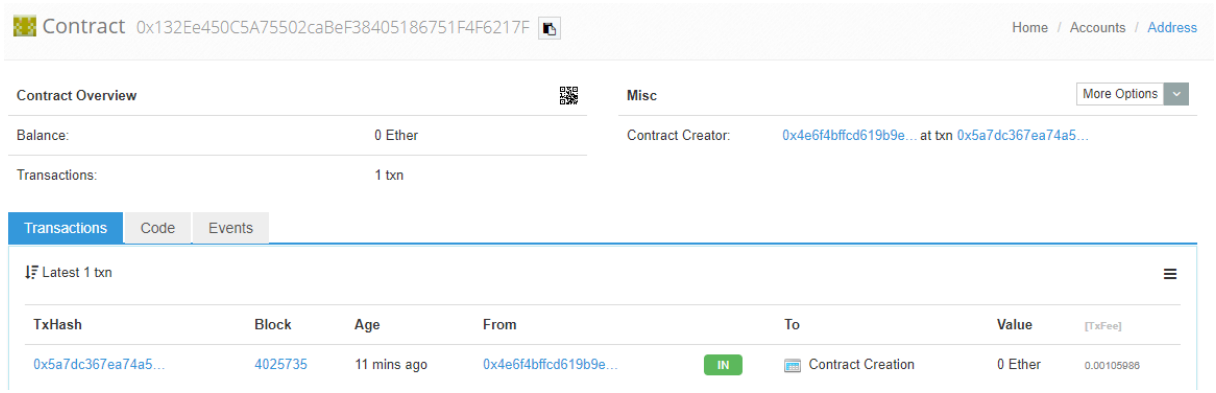


Ilustración 27 Contrato desplegado en Ropsten (Etherscan, 2018)

En el área de contenido del contrato '***Deployed Contracts***' procedemos a verificar las opciones que existen para la utilización y realización de transacciones del Token creado, en el mismo se puede realizar la transferencia de valores entre cualquier cuenta que maneje Ethereum y posea las claves privadas, para lo cual utilizaremos las cuentas creadas, '***Cuenta01, Cuenta 02. Cuenta 03***' dentro de MetaMask y realizaremos los pagos pactados para el proyecto. Los botones de color **celeste** sirven para realizar consultas y no tienen costos dentro de la blockchain, los botones de color **rosa** implican transacciones o cambios, los cuales implican un procesamiento en la EVM y se debe pagar un costo en GAS.

Ilustración 28 Contrato Desplegado (ethereum/remix, 2018)

La primera acción que se realiza es confirmar que las 70000 monedas que se han creado se hayan acreditado correctamente a la cuenta del creados "**01 MAIN**" para lo que se utiliza el campo '***balanceOf***' introducimos la Wallet de la cuenta y presionamos el botón, el cual

despliega la cantidad de Tokens que posee. Y de igual manera procedemos a verificar el saldo de las 3 cuentas restantes, verificando en la primera los 70000 Tokens y en las 3 cuentas el valor de 0 Tokens.

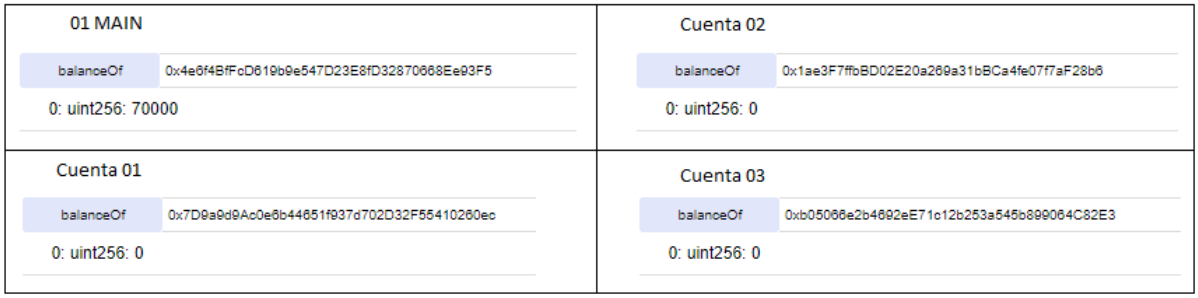


Ilustración 29 Saldos Iniciales Cuentas (ethereum/remix, 2018)

A continuación se procede a realizar las transferencias en el supuesto caso de el pago del empleador a sus empleados un sueldo total de $5000 valorados en 5000 Tokens E-USA, para lo cual se asume que la **Cuenta 01** recibe E-USA 2500 Tokens, la **Cuenta 02** recibe E-USA 1500 y la **Cuenta 03** recibe E-USA 1000 Tokens.

Para lo cual todas las transacciones quedan grabadas en la blockchain, las mismas que no pueden ser modificadas o adulteradas en un futuro, con lo cual se aplica el principio del '***No repudio***', presente en la criptografía.

Para realizar las transferencias de los Tokens, se debe tener en cuenta que, en el área de ejecución del código, se encuentre seleccionada la cuenta que contiene los fondos que se van a gastar o transferir, por el método de pruebas se realiza en ambiente solidity para comprobar funcionalidad a futuro, se puede crear en base a la web3.js y combinación de HTML, una aplicación o página web que nos permita realizar la transferencias necesarias en ambiente gráfico para el usuario final.

Se procede a realizar la acreditación de los sueldos a cada cuenta:

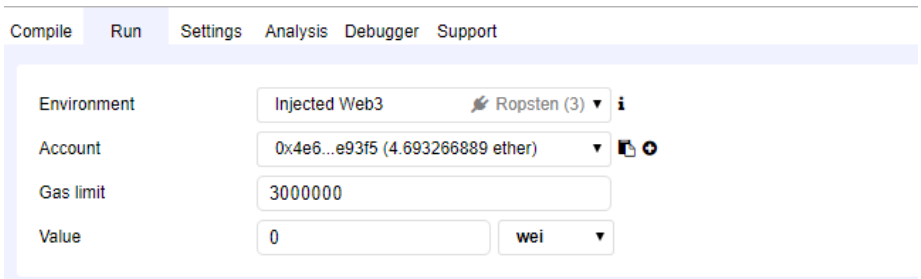
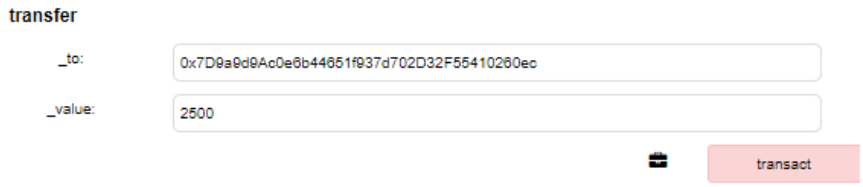


Ilustración 30 Cuenta origen de los Tokens (ethereum/remix, 2018)

Procedemos con la primera transacción desde "**01 MAIN**" 2500 Tokens a la "**Cuenta 01**"



Al procesar la transacción, nuevamente se envía una difusión a todos los mineros de que se confirme la acción que se resten los fondos de la cuenta origen y se acrediten los fondos a la cuenta de destino, para lo cual se deben pagar los costos en GAS, y aceptando enviar desde la billetera de MetaMax de la siguiente manera:

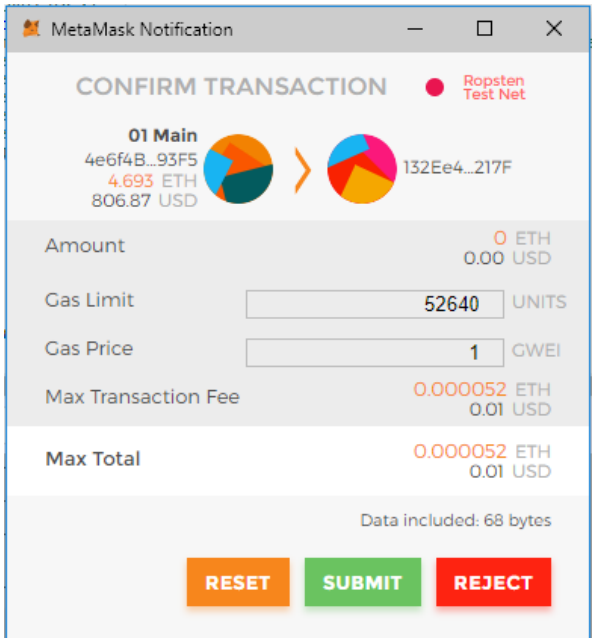


Ilustración 31 Transacción 1 MetaMax (Chrome Webstore, 2018)

La presente transacción luego de ser minada e ingresada en un bloque queda registrada dentro de la blockchain. Dentro de la web https://ropsten.etherscan.io consultamos la dirección de la Wallet de la "Cuenta 01" 0x7d9a9d9ac0e6b44651f937d702d32f55410260ec.

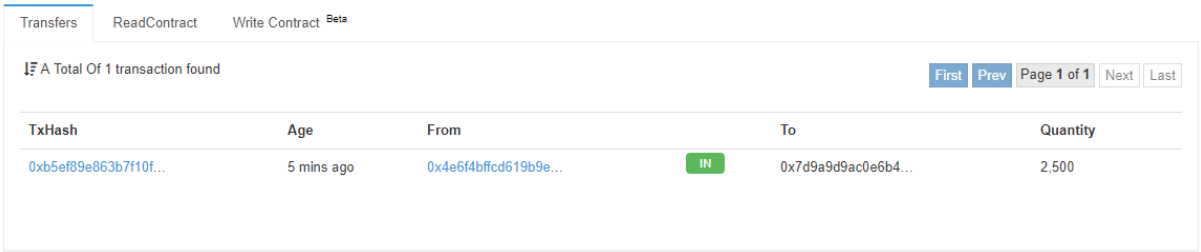


Ilustración 32 Transacción 1 en Blockchain (Etherscan, 2018)

De la misma manera procedemos con las 2 transacciones restantes.

Al finalizar las acreditaciones de los sueldos, podemos verificar los sueldos acreditados en cada cuenta.

Ilustración 33 Saldos Finales Cuentas (ethereum/remix, 2018)

En la cuenta principal creadora del contrato y tenedora original de los Tokens se puede verificar toda la información relacionada con los movimientos, además se puede hacer la trazabilidad de a donde fueron y de donde vinieron todas las transacciones, razón por la cual, ésta implementación en un nivel publico / privado, brinda la seguridad y transparencia de todos y cada uno de los movimientos realizados, evitando el cometimiento de corrupción o delitos informáticos, dolo, enriquecimiento ilícito, defraudación estatal, entre muchos otros casos.

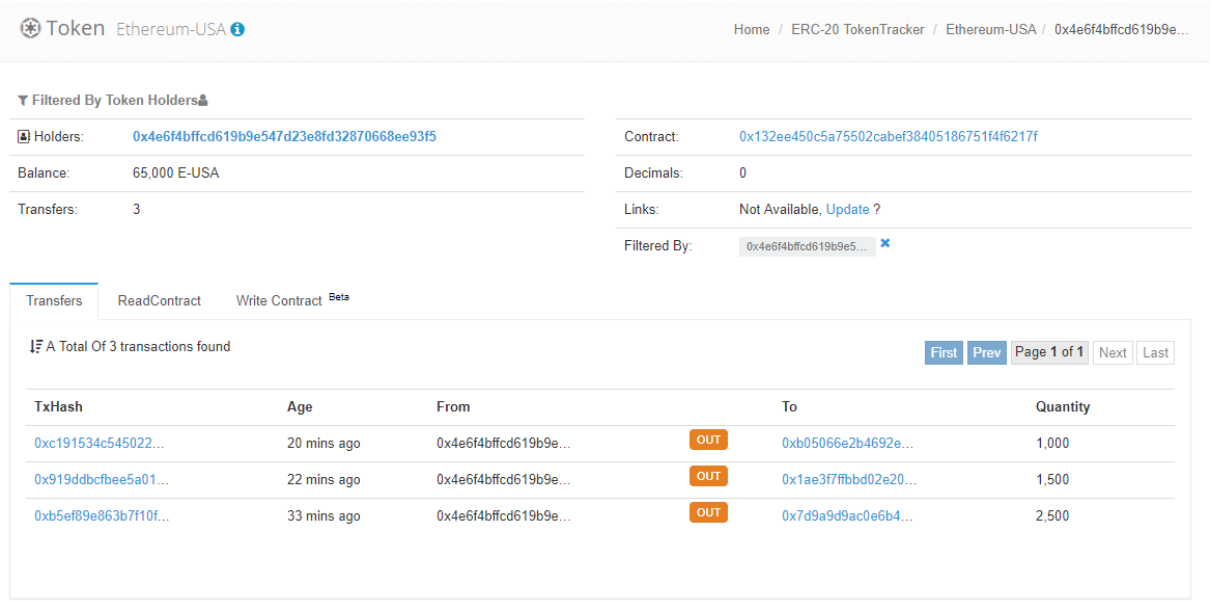


Ilustración 34 Estado de Movimientos "01 MAIN" (Etherscan, 2018)

Los Tokens pueden ser agregados a cualquier cuenta Ethereum alrededor del planeta, se pueden mostrar siguiendo unos sencillos pasos que se resumen a continuación:

- Enviar los Tokens a cualquier dirección Ethereum.
- Dentro de las billeteras Ethereum autorizadas (Ethereum Wallet, Mist, MyEtherWallet, MetaMax, etc.), se procede a agregar el Token, ingresando el número del Smart Contract, automáticamente aparecerán los datos correspondientes al Token, y finalmente se mostrará el saldo en Tokens que le corresponden a esa dirección criptográfica de Ethereum.

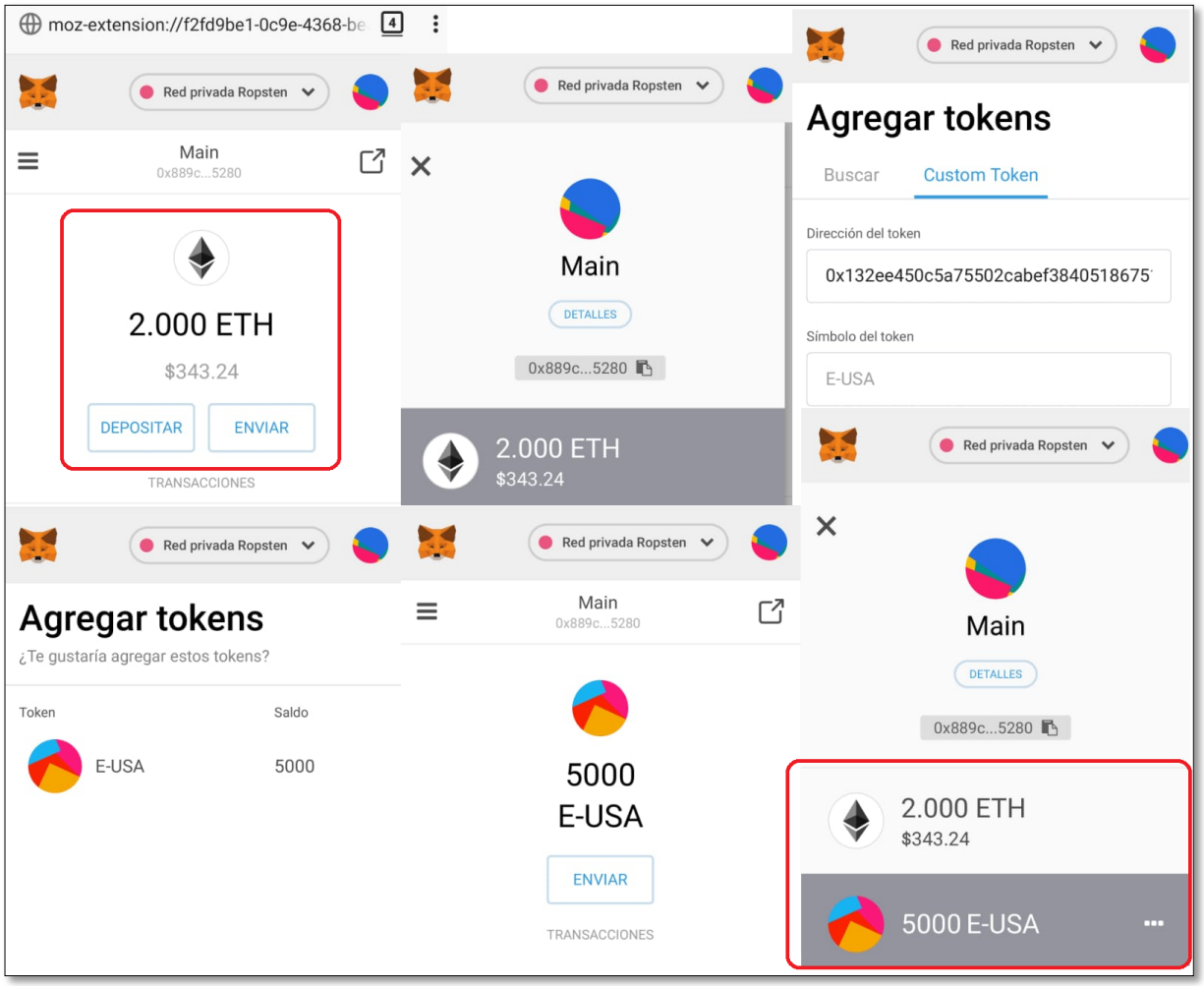


Ilustración 35 Agregar Token a Wallet Ethereum (MetaMax, 2018)

En la ilustración 35 se muestra como otra cuenta en otro dispositivo ha sido cargada con 5000 Tokens E-USA, reduciendo el monto de la cuenta original que mantenía todos los Tokens.

4.4. Descripción del piloto experimental

En el presente trabajo se propone la creación de un Token por medio de contratos inteligentes, el uso de las herramientas Solidity y Remix, herramientas online, que no requieren de instalación de programas propietario, como es el caso de Truffle, en que se necesita la pre - instalación de software avanzado como lo es Microsoft Visual Studio, levantar máquinas virtuales en Microsoft Azure, e incluso la utilización para test, es necesaria la cuenta de prueba que brindan de obsequio $200 para el desarrollo de equipos virtuales y similares, con lo que terminado el saldo o el periodo de prueba, seguir probando podría tener un costo elevado.

Con la anterior información se procede al desarrollo de una criptomoneda un TokenERC20 ‘Ethereum-USA’ con su símbolo a utilizar E-USA, que tiene un saldo inicial de 200.000,00 unidades y con el manejo de 2 decimales, para lo cual se ha seguido la presente técnica que permitirá realizar las siguientes acciones principales:

- Verificar la versión del código fuente, con el fin de que el código del contrato escrito no se comporte de manera diferente a lo esperado. Para el presente ejercicio que se puede duplicar dentro de la TestNet de ROPSTEN, la versión pragma solidity ^0.4.22 nightly.2018.4.4

- Siempre verificar que el contrato cumpla con los parámetros de la guía de convención para escritura de contratos inteligentes en solidity adjunta en el Anexo 4, (Ethereum, 2017)

- Si el contrato no es de creación de un Token o de criptoactivos, finalizar los contratos con la función selfdestruct, para que una vez terminados no permanezcan utilizando espacio en la cadena de bloques.

- Declarar variables públicas y privadas, dependiendo de la visibilidad que necesite cada cuenta del contrato inteligente, en algunos casos si no se declaran como publicas los contratos las asumen privadas y otros proyectos que necesitemos que escuchen nuestras variables no lo podrán hacer.

- Recordar que todas las transacciones en Solidity tienen un costo por su despliegue, solo las transacciones de consulta no tienen costo, las transacciones que realizan cambios tienen costos en GAS, éste costo de GAS no se puede hacer un reverso, si la transacción no se cumple por algún mal funcionamiento o programación el costo de GAS utilizado por esa transacción ‘***no exitosa***’, no se devolverá y además la transacción no se realizará.

- El lenguaje de Solidity que se ejecuta sobre la red Ethereum es un lenguaje de alto nivel, es decir es Turing complete, que permite programar prácticamente todo lo que deseemos crear, la diferencia esencialmente radica en que no se puede almacenar archivos, solamente los resultados de las funciones de hash utilizados en la comprobación del archivo, en base a las firmas digitales y sus resultados de las operaciones criptográficas, para solucionar el problema del almacenamiento de archivos se está recurriendo a la tecnología IPFS detallada dentro del estado del arte.

- A continuación se describe la forma en que se aplica la metodología de la creación de los contratos inteligentes aplicando a la creación de un TokenERC20, **enfatizando con negrilla** (Con el fin de resaltar dentro de las líneas de código), los pasos que no deben dejarse de lado o que deben ser indispensables en la creación de cualquier moneda electrónica (Ethereum Foundation, 2018), basada en esta tecnología.

- El presente contrato inteligente sirve para:

    a. Crear la criptomoneda circulante la misma que se acreditara a una cuenta de creadora del contrato.
    b. Transferir criptomonedas a diferentes cuentas.
    c. Quemar o eliminar criptomonedas.
    d. Comprar y vender monedas.
    e. Bloquear y desbloquear cuentas o address.

```
1.  pragma solidity ^0.4.21; // Verificar la versión del código 0.4.22 nightly.2018.4.4
2.
3.  contract owned {  // Sirve para cambiar el dueño del contrato
4.     address public owner;
5.
6.     function owned() public { // Ejecuta la función del contrato
7.         owner = msg.sender;   // verifica el propietario
8.     }
9.
10.    modifier onlyOwner {   // permite que solo el propietario modifique el contrato
11.        require(msg.sender == owner); // verifica que sea exactamente el propietario
12.        _;                            // si cumple con lo requerido, continúa
13.    }
14.
15.    function transferOwnership(address newOwner) onlyOwner public {  // cambia de propietario
16.    owner = newOwner;        // ratifica el nuevo propietario del contrato que puede modificar
17.    }
18. }
19.
20. interface tokenRecipient { function receiveApproval(address _from, uint256 _value, address
    _token, bytes _extraData) external; }
21.
22. contract TokenERC20 {
23.     // Variables públicas del Token
24.     string public name;
25.     string public symbol;
```

```
26.     uint8 public decimals = 18;
27.     // 18 decimales es muy recomendado, evite cambiarlo
28.     uint256 public totalSupply;
29.
30.     // Esto crea un array con todos los cambios
31.     mapping (address => uint256) public balanceOf;
32.     mapping (address => mapping (address => uint256)) public allowance;
33.
34.     // Esto genera un evento público en el blockchain que notificará a los clientes
35.     event Transfer(address indexed from, address indexed to, uint256 value);
36.
37.     // Esto genera un evento público en el blockchain que notificará a los clientes
38.     event Approval(address indexed _owner, address indexed _spender, uint256 _value);
39.
40.     // Esto notifica a los clientes sobre la cantidad quemada
41.     event Burn(address indexed from, uint256 value);
42.
43.     /*
44.     * Función Constructor
45.     *
46.     * Inicializa el contrato con los Tokens de oferta inicial al creador del contrato
47.     */
48.
49.     function TokenERC20(
50.         uint256 initialSupply,
51.         string tokenName,
52.         string tokenSymbol
53.     ) public {
54.         totalSupply = initialSupply * 10 ** uint256(decimals); // Actualiza el suministro
    total con la cantidad decimal
55.         balanceOf[msg.sender] = totalSupply; // Dar al creador todos los Tokens iniciales
56.         name = tokenName;               // Establece el nombre para fines de visualización
57.         symbol = tokenSymbol;           // Establece el símbolo para fines de visualización
58.     }
59.
60.     /*
61.     Transferencia interna, solo se puede invocar mediante este contrato
62.     */
63.     function _transfer(address _from, address _to, uint _value) internal {
64.         // Impedir la transferencia a la dirección 0x0. Use burn () en su lugar
65.         require(_to != 0x0);
66.         // Verifica si el remitente tiene suficientes Tokens
67.         require(balanceOf[_from] >= _value);
68.         // Verificar desbordamientos
69.         require(balanceOf[_to] + _value > balanceOf[_to]);
70.         // Guarde esto para una confirmación en el futuro
71.         uint previousBalances = balanceOf[_from] + balanceOf[_to];
72.         // Resta del Remitente
73.         balanceOf[_from] -= _value;
74.         // Agregue el mismo valor al destinatario
75.         balanceOf[_to] += _value;
76.         emit Transfer(_from, _to, _value);
77.         // Los Asserts son utilizados para análisis estático para encontrar errores en su
    código.Nunca deberían fallar
78.         assert(balanceOf[_from] + balanceOf[_to] == previousBalances);
79.     }
80.
81.     /*
82.     * Transferencia de Tokens
83.     *
84.     * Enviar `_value` Tokens a `_to` desde su cuenta
85.     *
86.     * @param _to La dirección del destinatario
87.     * @param _value La cantidad a enviar
88.     */
```

```
89.
90.     function transfer(address _to, uint256 _value) public returns (bool success) {
91.     _transfer(msg.sender, _to, _value);
92.     return true;
93.     }
94.
95.     /*
96.     * Transferir Tokens desde otras direcciones
97.     *
98.     * Enviar `_value` Tokens a `_to` en nombre de `_from`
99.     *
100.    * @param _from La dirección del remitente
101.    * @param _to La dirección del destinatario
102.    * @param _value la cantidad a enviar
103.    */
104.
105.    function transferFrom(address _from, address _to, uint256 _value) public returns (bool
    success) {
106.        require(_value <= allowance[_from][msg.sender]);   // Verifique el valor
107.        allowance[_from][msg.sender] -= _value;
108.        _transfer(_from, _to, _value);
109.        return true;
110.    }
111.
112.    /*
113.    * Establecer la asignación para otra dirección
114.    *
115.    * Permite `_spender` no gastar más de `_value` Tokens en tu nombre
116.    *
117.    * @param _spender La dirección autorizada para gastar
118.    * @param _value La cantidad máxima que puede gastar
119.    */
120.
121.    function approve(address _spender, uint256 _value) public
122.        returns (bool success) {
123.        allowance[msg.sender][_spender] = _value;
124.        emit Approval(msg.sender, _spender, _value);
125.        return true;
126.    }
127.
128.    /*
129.    * Establecer la asignación para otra dirección y notificar
130.    */
131.    * Permite `_spender` no gastar más de `_value` Tokens en tu nombre, y luego notificar al
    contrato acerca de eso
132.    /*
133.    * @param _spender La dirección autorizada para gastar
134.    * @param _value la cantidad máxima que pueden gastar
135.    * @param _extraData información adicional para enviar al contrato aprobado
136.    */
137.
138.    function approveAndCall(address _spender, uint256 _value, bytes _extraData) public returns
    (bool success) {
139.        tokenRecipient spender = tokenRecipient(_spender);
140.        if (approve(_spender, _value)) {
141.        spender.receiveApproval(msg.sender, _value, this, _extraData);
142.        return true;
143.        }
144.    }
145.
146.    /*
147.    * Destruye Tokens
148.    *
149.    * Eliminar `_value` Tokens del sistema de forma irreversible
150.    *
```

```
151.    * @param _value la cantidad de Tokens para quemar
152.    */
153.
154.    function burn(uint256 _value) public returns (bool success) {
155.        require(balanceOf[msg.sender] >= _value);   // Verifica saldo de remitente
156.        balanceOf[msg.sender] -= _value;            // Resta del remitente
157.        totalSupply -= _value;                      // Actualiza totalSupply
158.        emit Burn(msg.sender, _value);
159.        return true;
160.    }
161.
162.    /*
163.    * Destruye Tokens desde otra cuenta
164.    *
165.    * Eliminar `_value` Tokens del sistema de forma irreversible a nombre de `_from`.
166.    *
167.    * @param _from la dirección del remitente
168.    * @param _value la cantidad de dinero para quemar
169.    */
170.
171.    function burnFrom(address _from, uint256 _value) public returns (bool success) {
172.        require(balanceOf[_from] >= _value);      // Verifica el saldo objetivo es suficiente
173.        require(_value <= allowance[_from][msg.sender]);  // Verifica saldo de remitente
174.        balanceOf[_from] -= _value;                       // Resta del saldo objetivo
175.        allowance[_from][msg.sender] -= _value;           // Resta del subsidio del remitente
176.        totalSupply -= _value;                            // Actualiza totalSupply
177.        emit Burn(_from, _value);
178.        return true;
179.        }
180.    }
181.
182.    /*
183.    |******************************************|
184.    |     EL TOKEN AVANZADO COMIENZA AQUÍ        |
185.    |******************************************| */
186.
187.contract EUSA is owned, TokenERC20 {
188.
189.    uint256 public sellPrice;
190.    uint256 public buyPrice;
191.
192.    mapping (address => bool) public frozenAccount;
193.
194.    /* Esto genera un evento público en el blockchain que notificará a los clientes */
195.    event FrozenFunds(address target, bool frozen);
196.
197.    /* Inicializa el contrato con los Tokens de oferta inicial al creador del contrato */
198.    function EUSA(
199.        uint256 initialSupply,
200.        string tokenName,
201.        string tokenSymbol
202.    ) TokenERC20(initialSupply, tokenName, tokenSymbol) public {}
203.
204.    /* Transferencia interna, solo se puede invocar mediante este contrato */
205.    function _transfer(address _from, address _to, uint _value) internal {
206.        require (_to != 0x0);        // Evitar transferir a dirección 0x0. Use burn () en vez
207.        require (balanceOf[_from] >= _value);  // Verifica si el remitente tiene suficiente
208.        require (balanceOf[_to] + _value >= balanceOf[_to]);  // Verificar desbordamientos
209.        require(!frozenAccount[_from]);          // Verifica si el remitente está congelado
210.        require(!frozenAccount[_to]);            // Verifica si el destinatario está congelado
211.        balanceOf[_from] -= _value;              // Resta del remitente
212.        balanceOf[_to] += _value;                // Agregue lo mismo al destinatario
213.        emit Transfer(_from, _to, _value);
214.    }
215.
```

```
    /*
    /// @notice Crea tokens `mintedAmount` y envíalo a` target`
    /// @param address target de destino para recibir los Tokens
    /// @param mintedAmount La cantidad de tokens que se recibirán
    */

    function mintToken(address target, uint256 mintedAmount) onlyOwner public {
        balanceOf[target] += mintedAmount;
        totalSupply += mintedAmount;
        emit Transfer(0, this, mintedAmount);
        emit Transfer(this, target, mintedAmount);
    }

    /*
    /// @notice `freeze? Prevent | Allow` `target` permitir o negar que envíe y reciba Tokens
    /// @param address target de destino para ser congelada
    /// @param freeze Para congelar o descongelar
    */

    function freezeAccount(address target, bool freeze) onlyOwner public {
        frozenAccount[target] = freeze;
        emit FrozenFunds(target, freeze);
    }

    /*
    /// @notice Permitir a los usuarios comprar Tokens para `newBuyPrice` eth y vender Tokens
    para `newSellPrice` eth
    /// @param newSellPrice Precio que los usuarios pueden vender según el contrato
    /// @param newBuyPrice Price los usuarios pueden comprar del contrato
    */

    function setPrices(uint256 newSellPrice, uint256 newBuyPrice) onlyOwner public {
        sellPrice = newSellPrice;
        buyPrice = newBuyPrice;
    }

    /// @notice Buy Compre Tokens enviando Ether

    function buy() payable public {
        uint amount = msg.value / buyPrice;               // Calcula el valor
        _transfer(this, msg.sender, amount);              // Realiza las transferencias
    }

    /*
    /// @notice Sell `amount` Vende los Tokens al contract
    /// @param amount cantidad de Tokens a ser vendidos
    */

    function sell(uint256 amount) public {
        address myAddress = this;
        require(myAddress.balance >= amount * sellPrice);  // Verifica si el contrato tiene
    suficientes Ether para comprar
        _transfer(msg.sender, this, amount);               // Realiza las transferencias
        msg.sender.transfer(amount * sellPrice);           // Envía Ether al vendedor. Es
    importante realizar este procedimiento al final para evitar ataques de repetición
        }
    }
```

Con el presente contrato inteligente programado su código en solidity, el cual es una base para la creación de cualquier TokenERC20, con un nivel de seguridad probado y recomendado, además existen otros contratos simples para la creación de un TokenERC20

en alrededor de 20 líneas de código '***como se puede observar en el Anexo 5'***, en efecto, se puede crear de esa manera, pero no es la recomendada ni la correcta, ya que no ofrece las funcionalidades necesarias ni evaluaciones de seguridad mínimas permitidas para el correcto desempeño.

Para la escritura y lectura de los Tags y su significado se tomara en cuenta la siguiente tabla:

Tabla 7 Tags de notación en Solidity para uso en comentarios

Tag	Description	Context
`@title`	A title that should describe the contract	contract, interface
`@author`	The name of the author of the contract	contract, interface, function
`@notice`	Explain to a user what a function does	contract, interface, function
`@dev`	Explain to a developer any extra details	contract, interface, function
`@param`	Documents a parameter just like in doxygen (must be followed by parameter name)	function
`@return`	Documents the return type of a contract's function	function

Adaptación de (GitHub, Inc., 2018)

4.5. Discusión

El presente piloto se encuentra desplegada en la red de pruebas de Ethereum en la TestNet de Ropsten, la misma que puede ser consultada a lo largo del tiempo, estos datos quedarán permanentemente, mientras exista un nodo de la red levantado, por tal motivo, la seguridad informática aplicada al experimento, queda demostrada aplicando las principales características que a continuación se detallan:

- **CONFIDENCIALIDAD**, al enviar las transacciones directamente a los usuarios que los han requerido.

- **INTEGRIDAD**, se da al no poder modificar sus datos, la capa de seguridad que imprime la criptografía de clave pública y privada dentro de cada sistema blockchain.

- **DISPONIBILIDAD**, se cumple al momento en que la red Ethereum (o cualquier red de la blockchain) se encuentran respaldado por los nodos P2P, que se encuentran distribuidos a nivel mundial y estarán disponibles las 24 horas, los 365 días del año.

- **NO REPUDIO**, ésta acción implica que las personas o usuarios que son anclados a las cuentas o wallet's, no pueden negar haber recibido la información o los valores transmitidos en las transacciones.

- **AUTENTICACION**, todas las cuentas, wallet's, billeteras, contratos, cuentas de Exchange, requieren un fuerte factor de autenticación a tal punto que una vez perdidas las claves de acceso, los fondos relacionados con estas cuentas pueden quedar perdidos de por vida sin acceso a nadie.

Todos estos aspectos pueden conformar sistemas extremadamente seguros y transparentes, donde el único límite es la inventiva del ser humano, todos los días se crean aplicaciones basadas en blockchain que basadas en la seguridad pueden ayudar a facilitar los acontecimientos de la vida diaria.

El presente piloto se puede dar en un contexto real, por ejemplo financiando al crear un fondo de inversión que pueda contener el monto real en dinero en una cuenta o en un fideicomiso que responda exactamente a los valores iniciales del contrato, que para el ejemplo fueron 70.000 unidades, en el caso de que se hubiese recogido el 50% de los fondos reales, se puede hacer una "quema" de los 35.000 Tokens que no están respaldados, de la misma manera que se realizaron las transacciones, para lo cual se demostrará de la siguiente manera:

Se utiliza la transacción '**burn**' la misma que recibe los Tokens y los quema, dejándolos fuera de todo el sistema, es literalmente como quemar los fajos de billetes físicamente.

Ilustración 36 Quema de 50% de Tokens (ethereum/remix, 2018)

La transacción queda registrada en la blockchain y los Tokens desaparecen, con esto se puede seguir trabajando con los Tokens respaldados físicamente por valores reales que concuerden en un contrato firmado electrónicamente, el mismo que emita un hash y este sea almacenado en la blockchain, o con un documento electrónico almacenado en el IPFS, el mismo que su hash será almacenado en la cadena de bloques, donde se verificará su autenticidad.

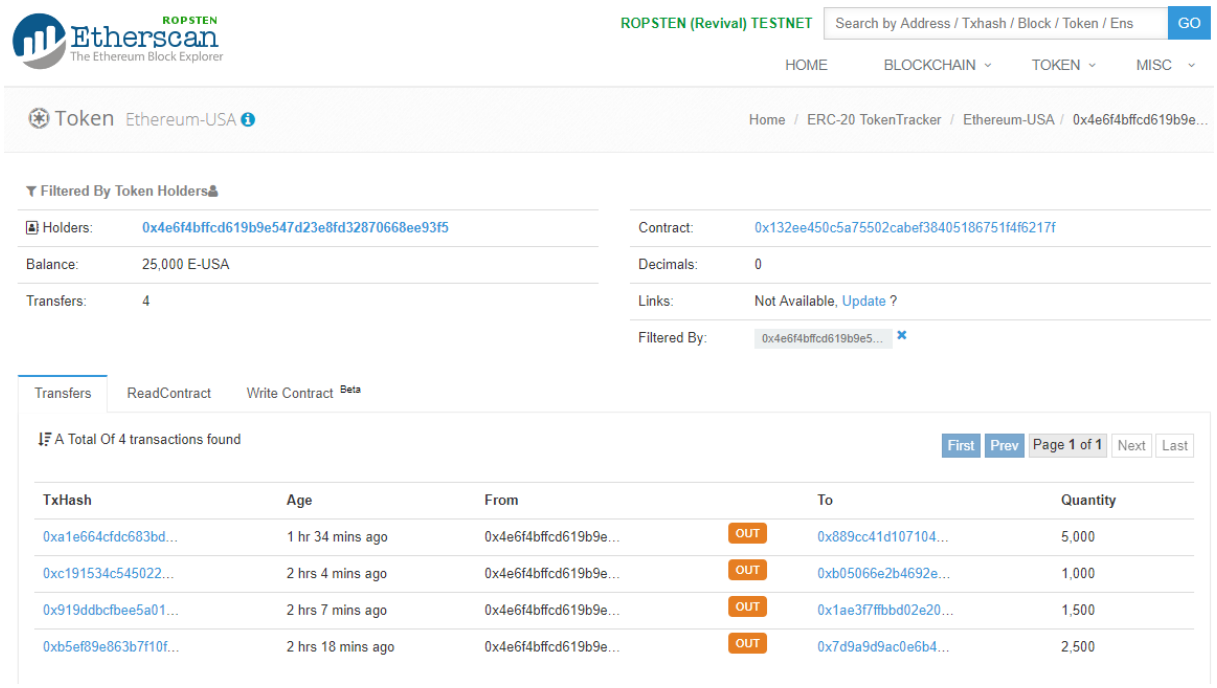


Ilustración 37 Estado Final cuenta "01 MAIN"

Al finalizar el experimento podemos observar los siguientes datos:

Monto original de Tokens 70.000

Transferencias enviadas 2.500, 1.500, 1.000, 5.000 = 10000 Tokens

Tokens quemados 35.000

Balance Final = 25.000 Tokens en la cuenta 01 MAIN y 10.000 en circulación.

Al concluir, las personas pueden realizar transacciones con otros usuarios y al final el dinero que se encuentra respaldado en el fideicomiso, se puede liquidar para convertir todos los Tokens en dinero en efectivo.

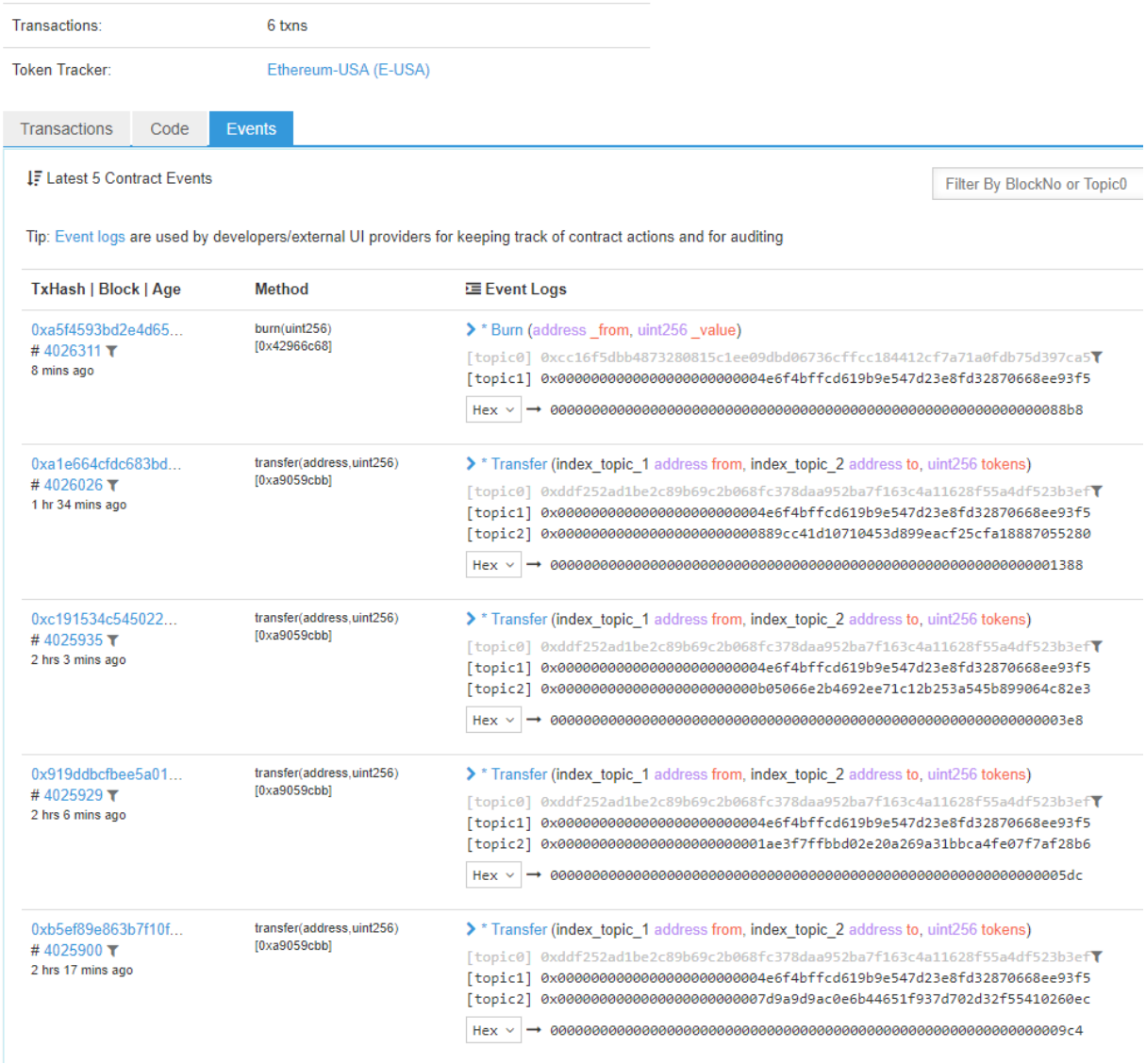


Ilustración 38 Transacciones realizadas en el Contrato (Etherscan, 2018)

Posteriormente los proyectos pueden crear aplicativos y billeteras exclusivamente para el uso de los Tokens, ingresarlos al sistema económico, añadiendo a locales o comercios donde se pueden comercializar, gastar, en compras y ventas o pagos de servicios. Todos los Tokens se encuentran respaldados y se conocerá el inicio o el fin de los movimientos de manera totalmente transparente.

5. Conclusiones y trabajo futuro

5.1. Conclusiones

Con este trabajo se ha tomado la opción de la creación de los contratos inteligentes por la vía más eficiente en cuanto a inversión en tiempo y conocimiento, se ha seleccionado el sistema de programación Solidity + Remix en el entorno web en conjunto con MetaMax, debido a que prácticamente solo se necesita un computador con internet y el conocimiento para aplicarlo, no necesita licenciamiento o instalación de programas de terceros para poder empezar a elaborar los contratos inteligentes. El conocimiento básico que se debe poseer es lenguaje HTML, CSS, JavaScript y el sistema blockchain.

La blockchain, las criptomonedas y los contratos inteligentes en la actualidad van de la mano, se puede crear pequeños sistemas financieros, que con el tiempo y la expansión de la tecnología relacionada pueden extenderse y llegar a ser sistemas regionales, nacionales y a nivel mundial, entre estos tenemos:

Control y Contabilidad interna de una pequeña o mediana empresa.

Core’s financieros contables en un ambiente local de instituciones financieras Pymes, con la ayuda de HyperLedger.

CrowdFounding (Micro mecenazgo), donde se puede enviar una idea de empresa y solicitar el financiamiento grupal o colectivo para llegar a dar cumplimiento a esas metas realizadas en un proyecto. (Artistas, músicos, solidaridad, apoyo a causas, etc.)

Rastreo de Origen. Confirmar el origen de todos los insumos que están inmersos en la creación de un determinado bien o servicio, y que toda la logística, procesos, intercambios, pagos, etc., queda registrada en la blockchain de manera inmutable.

Con la creación o introducción de IoT, en el caso de los vehículos, sería imposible modificar el recorrido del kilometraje al odómetro o velocímetro.

Registro de bienes muebles o inmuebles, creación de un registro de la propiedad, evitando las estafas en ventas duplicadas del mismo bien a varias personas (doble gasto).

Registro de identidad nacional, además del DNI electrónico que ya poseen algunos países, incluida España, y que poseen firmas digitales.

Con la correcta creación de las criptomonedas o Tokenizacion de activos sobre la blockchain y con el uso de los contratos inteligentes, se puede emitir una certificación de posesión de un bien y éste ser reconocido a nivel mundial, resolver o transparentar las acciones realizadas, o las que no se han realizado, por omisiones, y se puede saber dónde o cual fue la falla en el trabajo que se quiso realizar, además, que el implicado no se puede esconder o deslindar de las acciones ejecutadas, aplicando el ***No Repudio***, todas éstas acciones realizadas tienen un usuario que se ha ***Autenticado***, creando la ***Confidencialidad***, y llevando la ***Integridad*** hasta el límite de la verdad, todo esto se puede lograr gracias a las cualidades que nos brinda la seguridad de la tecnología blockchain, por lo visto y aplicado recomienda utilizar de manera correcta ésta tecnología basada en la ***criptografía***.

- Toda la información generada por este piloto se la puede encontrar buscando la dirección del contrato, o la dirección del creador del contrato, o por el nombre del Token E-USA dentro de las redes de blockchain de Ethereum TestNet de Ropsten:

Contrato

https://ropsten.etherscan.io/address/0x132ee450c5a75502cabef38405186751f4f6217f

Cuenta Principal

https://ropsten.etherscan.io/address/0x4e6f4bffcd619b9e547d23e8fd32870668ee93f5#tokentxns

5.2. Líneas de trabajo futuro

Al crear las criptomonedas que se encuentren ancladas a los valores de las monedas nacionales, se puede incluso '***tradear***' con ellas en Exchanges, además se puede proponer la creación propia de una moneda madre que contenga todas las criptomonedas basadas en anclajes a valores externos, se propone el nombre de la moneda '**E-WORLD**' (Ethereum-World) a manera de Bitcoin, esta moneda se capitalizaría en base a las ganancias que se obtengan de hacer **TRADING** o las comisiones de pago que se generan por las transferencias y la adopción de sus monedas hijas, en conclusión, a las monedas anteriormente expuestas que se encuentran anclados los valores a monedas nacionales, se propone '***Tokenizar***' todos

los activos de valor, anclándolos a su precio actual, es decir se pueden crear Tokens con el valor del Oro, plata, piedras preciosas, etc., también se propone la '***Tokenizacion***' de acciones de la bolsa por ejemplo, acciones de Facebook, Google, Amazon, etc.

Con esta acción lo que lograremos es la parte contraria de lo que se pretende hacer, que Bitcoin ingrese en los futuros de la bolsa de valores, '***Tokenizaremos***' la Bolsa de valores y traeremos sus cotizaciones en forma de Tokens directo al mundo de las Criptomonedas, indirectamente o de manera colateral, y lograr que los tenedores de criptomonedas, puedan interactuar con mercados bursátiles virtualizados por medio de Tokens.

Se podrá comercializar ORO o el Token que se encuentra anclado al valor del ORO '**E-GOLD**' (Ethereum-Gold), y tendría prácticamente el mismo efecto.

Con los datos anteriormente expuestos obtendremos una lista de posibles monedas que puedan representar valores a nivel mundial.

Tabla 8 Tokenizacion de Valores Reales

SIMBOLO	EQUIVALENCIA
E-GOLD	1 GRAMO DE ORO
E-SILVER	1 GRAMO DE PLATA
E-PLATINO	1 GRAMO DE PLATINO
E-RODIO	1 GRAMO DE RODIO
E-PETRO	1 BARRIL DE PETROLEO*
E-PLUTONIO	1 GRAMO DE PLUTONIO
E-DIAMOND	1 GRAMO DE DIAMANTE
E-FACE	1 ACCION DE FACEBOOK
E-GOOGLE	1 ACCION DE GOOGLE
E-AMAZON	1 ACCION DE AMAZON
E-APPLE	1 ACCION DE APPLE
E-BMW	1 ACCION DE BMW
E-MICROSOFT	1 ACCION DE MICROSOFT
E-TOYOTA	1 ACCION DE TOYOTA
E-FERRARI	1 ACCION DE FERRARI
E-LAMBO	1 ACCION DE LAMBORGHINI

Los pares de todas las monedas E-WORLD serían directamente tradeables o intercambiables por Ethereum ETH, debido que todas las cuentas de Ethereum aceptan Tokens ERC20, en conclusión se pueden realizar las transferencias teniendo pares como estos:

Tabla 9 Pares de Criptomonedas

E-USA / ETH
E-GOLD / ETH
E-PETRO / ETH

Un ejemplo de transacción seria la siguiente:

- Compra de 10 gramos de oro.

    a. Convertir los dólares en criptomonedas, en este caso ETH.
        i. Se puede acceder a nivel mundial a la página Localbotcoins.com explicado con anterioridad, o se puede también utilizar localethereum.org que tiene el mismo fin.
        ii. Realizar el depósito y confirmar la transacción en la página web.
        iii. Obtener las criptomonedas.

    b. Convertir los ETH a ORO
        i. Ingresar al Exchange que tenga listada la moneda E-GOLD
        ii. Realizar el depósito de ETH al Exchange
        iii. Cambiar ETH por E-GOLD

    c. Almacenar / vender cuando se necesite o para crear ganancias.
        i. Mantener el valor en el Exchange, o,
        ii. Retirar la moneda E-GOLD para almacenarla en su monedero, en el cual se posean las claves privadas para tener mayor seguridad.

    d. Convertir los E-GOLD en efectivo
        i. Transferirlos de la Wallet al Exchange
        ii. Cambiar en el Exchange los E-GOLD por ETH
        iii. Vender los ETH en localethereum.org, o,
        iv. Cambiar los ETH por BTC en localbitcoins.com
        v. Recibir el Efectivo

Todo esto se puede realizar al Tokenizar un Activo de valor.

6. Bibliografía

Bibliografía

Androutsellis, S., Spinellis, T., & Spinellis, D. (2004). A Survey of Peer-to-Peer Content Distribution Technologies. En A. C. Surveys. New York.

Angulo, S. (25 de Abril de 2016). *www.enter.co*. Obtenido de http://www.enter.co/chips-bits/seguridad/hackearon-el-sistema-bancario-swift/

API docs. (1 de Agosto de 2018). *https://faucet.ropsten.be/*. Obtenido de https://faucet.ropsten.be/

Bergquist, J. H. (30 de Junio de 2017). Blockchain Technology and Smart Contracts - Privacy-preserving Tools. Upsala, Suecia.

bitaddress.org. (22 de Septiembre de 2017). *https://bitcoinpaperwallet.com.* Obtenido de https://bitcoinpaperwallet.com/bitcoinpaperwallet/generate-wallet.html#

bitaddress.org. (2018). *https://www.bitaddress.org/.* Obtenido de https://www.bitaddress.org/bitaddress.org-v3.3.0-SHA256-dec17c07685e1870960903d8f58090475b25af946fe95a734f88408cef4aa194.html

BitcoinMagazine. (s.f.). *BitcoinMagazine*. Obtenido de https://bitcoinmagazine.com/articles/yes-bitcoin-can-do-smart-contracts-and-particl-demonstrates-how/

bitinfocharts@gmail.com. (2018). *https://bitinfocharts.com.* Obtenido de https://bitinfocharts.com

BITMAIN. (2018). *https://shop.bitmain.com/.* Obtenido de https://shop.bitmain.com/promote/antminer_s9i_asic_bitcoin_miner/overview?flag=overview

Blockchain.com. (25 de Agosto de 2018). *Blockchain.com*. Obtenido de https://www.blockchain.com/es/charts/market-price

blockchain.info. (2017). *blockchain.info*. Obtenido de https://blockchain.info/es/tree/283858998

Chrome Webstore. (2018). *https://chrome.google.com/webstore/*. Obtenido de https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn/related?hl=es

coinmarketcap.com. (2018). *https://coinmarketcap.com.* Obtenido de https://coinmarketcap.com/currencies/filecoin/

CoinWarz. (2018). *https://www.coinwarz.com.* Obtenido de https://www.coinwarz.com

CRIPTOTENDENCIA. (16 de Junio de 2017). *https://criptotendencia.com*. Obtenido de https://criptotendencia.com/2017/06/26/que-son-los-tokens-erc20/

CuriousInventor. (14 de 07 de 2013). *youtube.com*. Obtenido de https://www.youtube.com/watch?v=Lx9zgZCMqXE&feature=youtu.be

Dead Coins. (2018). *deadcoins.com.* Obtenido de http://deadcoins.com

Ethereum. (2017). *https://solidity-es.readthedocs.io*. Obtenido de https://solidity-es.readthedocs.io/es/latest/style-guide.html

Ethereum Foundation. (2018). *https://www.ethereum.org*. Obtenido de https://www.ethereum.org/token

ethereum/remix. (08 de 2018). *http://remix.ethereum.org.* Obtenido de http://remix.ethereum.org/#optimize=true&version=soljson-v0.4.21+commit.dfe3193c.js

EthereumEng. (2016). *https://eleanarey.github.io*. Obtenido de https://eleanarey.github.io/solidity-en-espa%C3%B1ol/

Ethermine. (20 de 05 de 2018). *https://www.ethermine.org.* Obtenido de https://www.ethermine.org/miners/e2928a7cb8165fe4e083fe93f481c38febf13396/dashboard

Etherscan. (2018). *https://ropsten.etherscan.io*. Obtenido de https://ropsten.etherscan.io/tx/0x5a7dc367ea74a546aeff4eb4c3e53ac601f3052d03b5363210427d2d09e820fc

Flores, R. (05 de Junio de 2017). *www.urgentebo.com*. Obtenido de http://www.urgentebo.com/noticia/reportaje-la-estafa-de-bitcoin-cash-afect%C3%B3-al-menos-100-mil-personas-en-bolivia

Gil, S. (2015). *http://economipedia.com*. Obtenido de http://economipedia.com/definiciones/swift.html

GitHub, Inc. (2018). *https://github.com/*. Obtenido de https://github.com/ethereum/wiki/wiki/Ethereum-Natural-Specification-Format

hoekomikaangeld.com. (Marzo de 2017). Obtenido de http://hoekomikaangeld.com/wp-content/uploads/2017/03/gladiacoin-team-network-1.png

Hollingworth, D. (3 de Julio de 2017). *https://www.pcauthority.com.au.* Obtenido de https://www.pcauthority.com.au/news/asus-release-amd-and-nvidia-powered-crypto-currency-mining-cards-467332

Hyperledger . (26 de Septiembre de 2016). *fabric-ericjl.readthedocs.io*. Obtenido de https://fabric-ericjl.readthedocs.io/es/release/index.html

James, R. (01 de Junio de 2018). *https://github.com/ethereum.* Obtenido de https://github.com/ethereum/wiki/wiki/%5BSpanish%5D-White-Paper.md

Juniper Research. (31 de Julio de 2017). *www.ibm.com.* Obtenido de https://www.ibm.com/blockchain/

Lara, I. (23 de Diciembre de 2017). *tecnops.es*. Obtenido de https://tecnops.es/3-ethereum-primeros-pasos-con-solidity-y-ethereum-javascript-api/

Medina, Y. (29 de Marzo de 2018). *Youtube.com.* Obtenido de https://www.youtube.com/watch?v=TJXFLWmnido&feature=youtu.be

MetaMax. (2018). *moz-extension*. Obtenido de moz-extension://f2fd9be1-0c9e-4368-bea2-fbdb6fce5cc2/popup.html#

MinerGate. (30 de 06 de 2018). *https://www.minergate.com.* Obtenido de https://www.minergate.com

mineXMR.com. (2018). *https://minexmr.com/.* Obtenido de https://minexmr.com/

Monax Industries. (31 de Mayo de 2018). *https://monax.io/*. Obtenido de https://monax.io/learn/smart_contracts/

Nakamoto, S. (2008). *https://bitcoin.org.* Obtenido de https://bitcoin.org/files/bitcoin-paper/bitcoin_es.pdf

Narayanan, A., Bonneau, J., Felten, E., Miller, A., Goldfeder, S., & Clark, J. (2016). *Bitcoin and Cryptocurrency Technologies.* Princeton: Princeton University Press.

OpenZeppelin. (2018). *https://openzeppelin.org/*. Obtenido de https://openzeppelin.org/

Ornubia Diaz, J. (13 de Diciembre de 2017). El Bitcoin, la nueva moneda virtual y su Ordenamiento jurídico. Madrid, España.

Pasten Lugo, H. A. (2012). *UNAM.* Obtenido de Sistema de Régimen de Inversión para Siefores , VaR y Notas Estructuradas con Opciones de Renta Variable: http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/2801/Tesis.pdf?sequence=1

PubliMetro. (22 de Junio de 2017). *www.metroecuador.com.ec*. Obtenido de https://www.metroecuador.com.ec/ec/colombia/2017/06/22/cayeron-dos-piramides-bitcoin-funcionan-colombia.html

Puigvert, M. (22 de Mayo de 2016). *criptonoticias.com*. Obtenido de https://criptonoticias.com/sucesos/conozca-historia-bitcoin-pizza-day-buen-provecho/

Ramírez Gracia, V. (08 de Febrero de 2018). Hacia un tratamiento uniforme del Bitcoin desde una perspectiva fiscal global. Sevilla, España.

Rojas, E. (5 de Febrero de 2015). *www.muycomputerpro.com*. Obtenido de http://www.muycomputerpro.com/2015/02/05/silk-road-mercado-on-line-drogas

Schüpfer, F. (15 de 08 de 2017). Design and Implementation of a Smart Contract Application. Lucern, Suiza.

Shares, D. (27 de Julio de 2016). *Bitcoin.com*. Obtenido de https://news.bitcoin.com/bitcoin-user-demographics-european-males-age-25-34/

Simply Explained. (14 de Mayo de 2018). *youtube.com*. Obtenido de https://www.youtube.com/watch?v=5Uj6uR3fp-U&feature=youtu.be

SofttekTV. (21 de Abril de 2017). *Youtube.com*. Obtenido de https://www.youtube.com/watch?v=TJXFLWmnido&feature=youtu.be

Sotomayor, L. (28 de Enero de 2012). *youtube.com*. Obtenido de https://www.youtube.com/watch?v=uaYUKR9ni7I

Topini, G. (28 de 11 de 2017). *bitcoinregs.org*. Obtenido de https://bitcoinregs.org/es/cuantas-personas-utilizan-bitcoin/

Tron, V. (14 de Diciembre de 2016). *blog.ethereum.org*. Obtenido de https://blog.ethereum.org/2016/12/15/swarm-alpha-public-pilot-basics-swarm/

Tron, V. (2018). *swarm-gateways.net*. Obtenido de http://swarm-gateways.net/bzz:/theswarm.eth/

Tur Faúndez, C. (2018). *SMART CONTRACTS Análisis Jurídico.* Madrid, España: REUS Editorial.

Wall, E., & Malm, G. (29 de Junio de 2016). Using Blockchain Technology and Smart Contracts to Create a Distributed Securities Depository. Lund, Suecia.

What to mine. (30 de 06 de 2018). *https://whattomine.com/.* Obtenido de https://whattomine.com/coins?utf8=%E2%9C%93&adapt_q_280x=1&adapt_q_380=0&adapt_q_fury=0&adapt_q_470=0&adapt_q_480=3&adapt_q_570=0&adapt_q_580=20&adapt_580=true&adapt_q_vega56=8&adapt_q_vega64=0&adapt_q_750Ti=0&adapt_q_1050Ti=0&adapt_q_10606=0&adapt_q_1070

Whitlock, M. (13 de Diciembre de 2012). *bitcointalk.org*. Obtenido de https://bitcointalk.org/index.php?topic=130619.0;all

Yuste Gonzalez, C. (Noviembre de 2015). Deep web y monedas virtuales: entorno privilegiado para las organizaciones terroristas.

ANEXOS

Anexo 1. Glosario de Términos

Para poder entender de mejor manera el mundo de las criptomonedas, existen términos diferentes a los usuales, por lo que se ha realizado un glosario de términos más utilizados con el fin de poder aclarar la lectura del documento.

ALTCOIN = Todas las monedas que nacieron a después de la creación del Bitcoin.

API = Application Programming Interface (Interfaz de Programación de Aplicaciones), se utiliza para obtener datos en tiempo real o programado, dentro de otra página o aplicación (web o móvil) que la haga referencia, por ejemplo, un widget para Android con la información del precio actual de Bitcoin cada 5 minutos.

BIFURCACIONES - FORKS

SOFTFORK. Su traducción al español es un TENEDOR SUAVE, lo cual no tiene nada que ver con el propio concepto, el SoftFork es una actualización del sistema que rige la criptomoneda, es decir, una actualización que sufre ligeros pero importantes cambios en la cual los miembros de la red pueden o no acogerse y el sistema sigue funcionando hasta cuando todos los miembros decidan si actualizar o no. Un ejemplo de este tipo de actualizaciones son las de un Sistema Operativo, por ejemplo, las Actualizaciones de Windows 7.

HARDFORK. Este representa una actualización a un nivel superior, donde los parámetros que se encuentran en el actual sistema no son compatibles con la nueva versión propuesta, en el caso del ejemplo anterior, se representaría como una actualización de Windows 7 hacia Windows 8 o 10.

El SoftFork y el HardFork en el caso de Bitcoin son el resultado de la aceptación de sugerencias de los usuarios, representadas por los BIP (Bitcoin Improvement Proposals – propuestas de mejora de Bitcoin), los más conocidos últimamente son el BIP 91 y el BIP 148.

SEGWIT. Segregated Witness (Testigo segregado o distribuido) Es una implementación al protocolo Bitcoin, que consiste en separar las firmas criptográficas que existen junto con el registro de las transacciones, dentro de un bloque de Bitcoin que es de 1MB con el objetivo de poder alojar más transacciones en el bloque y que los tiempos de espera en confirmación mejoren de 1 a 3 veces. Las firmas de las transacciones aparecerán adjuntas en otro sector fuera del bloque, pero irán ligados al mismo, esto es un SOFTFORK, o una actualización a bajo/medio nivel, que puede ser implementado al protocolo Bitcoin como una actualización.

SEGWIT2X. Es la aplicación del SegWit y además incrementar el tamaño de bloque de 1MB a 2MB, esto es un HARDFORK o una actualización de alto nivel, en la que los tamaños de bloque cambiaran desde el momento en que sea implementado, y si no hay consenso podría nuevamente existir una bifurcación, y con esto creando 2 monedas, como ocurrió el 1 de agosto de 2017, con el nacimiento de la criptomoneda BitcoinCash.

UASF = USER ACTIVATED SOFT FORK, activación de SoftFork impulsada por los usuarios.

UAHF = USER ACTIVATED SOFT FORK, activación de HardFork impulsada por los usuarios.

BIP = Bitcoin Improvement Proposals – Propuestas de mejora de Bitcoin.

BTU = Bitcoin Unlimited, grupo de mineros que desean cambiar las condiciones de Bitcoin, y que en la actualidad se dividió hacia BitcoinCash.

CLOUD MINING = Minería en la nube, grandes empresas rentan poder de cómputo para que se utilice esa fuerza de minado mediante contratos ya sean por tiempo límite o por tiempo indefinido, por lo general no son muy rentables y existen varias estafas y scam también en este modelo, las más confiables son: Genesis Mining, Hashflare y HashNest.

CÓDIGO QR = Código en forma rectangular, similar al código de barras, que indica un texto codificado para lectura especifica de dispositivos móviles, se utiliza para la lectura de direcciones de criptomonedas y no tener que insertar todo el resultado de la clave publica manualmente con el fin de evitar errores.

DDoS = Ataques de denegación de servicio, se conjugan cuando existen varias llamadas de solicitud de recursos a una web específica por parte de atacantes, con el objetivo de poner fuera de línea el servicio atacado.

DOBLE GASTO = (Double Spend) gastar dos veces el mismo dinero, pagar a 2 o más personas por diferentes servicios por un valor inexistente.

ECDSA = Elliptic Curve Digital Signature Algorithm, Algoritmo de firma criptográfica de curva elíptica, que se encuentra dividido en clave pública y privada.

ESCRITURA CAMELLO = Se utiliza en programación para juntar dos o más palabras para facilitar su lectura de la siguiente manera. JuntarDosPalabrasParaEntenderMejor

EXCHANGE = Centro de intercambio de criptodivisas, a especie de bolsa de valores.

FAUCET = Aplicación online que brinda recompensas en satoshis, Ethereum, o Ethereum para Test de aplicaciones, por cumplir o completar ciertos requerimientos, existen muchas que son scam o estafas.

FEES = comisiones que cobran las casas de cambio (Exchanges), los pools de minería, los sitios de conversión de criptodivisas sin necesidad de registro.

FUTUROS = Son operaciones en la Bolsa de valores donde se operan en base a especulaciones de compra y venta a tiempos pactados próximamente.

HASH = Intento de encontrar la respuesta a un algoritmo criptográfico (desafío matemático) y con este resultado poder anclar un nuevo bloque a la cadena de bloques.

H/s = 1 HASH por segundo

KILOHASH = KH/s = 1,000 HASHES por segundo 1,000 hashes

MEGAHASH = MH/s = 1,000 KILOHASHES por segundo 1,000,000 hashes

GIGAHASH = GH/s = 1,000 MEGAHASHES por segundo 1,000,000,000 hashes

TERAHASH = TH/s = 1,000 GIGAHASHES por segundo 1,000,000,000,000 hashes

HTML = Hyper Text Markup Language, lenguaje de marcas de hipertexto. Que sirve para el desarrollo de estructuras de páginas web.

CSS = Cascade Style Sheet, hojas de estilo en cascada. Sirven para dar forma y estilos a los documentos HTML, dentro de una App o página web.

ICO = Initial Coin Offering, (Oferta Inicial de Monedas), Lanzamiento de proyectos basado en criptomonedas donde se recaudan los fondos y se reparten monedas iniciales con precios favorables u otros beneficios, son llamados también CrowdFounding.

IOT = Internet of Things 'Internet de las cosas', visión futurista donde todos los artefactos poseen conexión a internet.

JSON = JavaScript Object Notation, es un formato para intercambio de datos, derivado de la notación literal de objetos de JavaScript, se utiliza como alternativa a XML o a las conocidas mallas.

NICK SZABO = Científico que propuso la creación de los Smart Contracts desde el año 1995, pero recién en el 2015 la infraestructura lógica y física se pueden ejecutar. Muchos creen que él puede ser el usuario del seudónimo Satoshi Nakamoto. Hace dos décadas creó es sistema monetario denominado BITGOLD.

NONCE = Cifra "comodín" que ayuda a obtener resultados en el cálculo de un hash especifico que asegura el anclaje de la cadena dentro de la Blockchain.

> "El "nonce" de un bloque Bitcoin es un campo de 32 bits (4 bytes) cuyo valor se establece de modo que el hash del bloque contenga una ristra de ceros. El resto de los campos no se pueden cambiar, ya que tienen un significado definido. Cualquier cambio en el bloque de datos (como pueden ser el nonce) hará que el hash del bloque sea completamente diferente. Dado que es imposible predecir qué combinación de bits se traducirá en el hash correcto, se intentan muchos nonce diferentes y el hash se vuelve a calcular para cada valor hasta que se encuentre un hash que contenga el número necesario de bits a cero. Como este cálculo iterativo requiere tiempo y recursos, la presentación del bloque con el valor nonce correcta constituye una prueba de trabajo." (Bitcoin Wiki, 2011)

El nonce es el único valor en una transacción que se puede cambiar para que la prueba de trabajo funcione, es similar a una prueba de abrir un candado en un llavero con 100 (o N) llaves diferentes, solo una de esas 100 llaves abrirá el candado, el nonce en este caso representaría cada llave utilizada para abrir el candado.

ORÁCULOS = Son entidades que brindan datos con capacidad de que Apis externas se conecten y puedan obtener resultados que sirvan de disparadores en la programación de Smart Contracts.

POW = Proof of Work, prueba de trabajo, resolución del problema matemático que permite verificar la veracidad de una transacción dentro de una blockchain publica como Bitcoin o Ethereum, conocido como minería.

POS = Proof of Stake, prueba de participación, otra forma de confirmar y resolver los problemas matemáticos basados en la prueba de trabajo pero de manera más ligera, y la recompensa está ligada a la cantidad de monedas que se posea.

POC = Proof of Concept, Prueba de Concepto, es un método propuesto para verificación de identidad

POOL = Nodo de una Blockchain a la cual se pueden afiliar los usuarios para cooperar en el proceso de minado colaborativo, prestando la potencia de su equipo.

PROTOCOLO = Es un conjunto de reglas que se establecen en el proceso de comunicación entre dos sistemas.

P2P / PEER TO PEER = También conocida como protocolo de comunicación P2P o punto a punto, es una red distribuida, en los que cada participante tiene una porción que es compartida en la red, para un texto más técnico citaré textualmente a (Androutsellis, Spinellis, & Spinellis, 2004) “Los sistemas P2P son sistemas distribuidos que consisten de nodos interconectados capaces de auto organizarse en topologías de red con el propósito de compartir recursos tales como contenido, ciclos de CPU, almacenamiento y ancho de banda, capaces de adaptarse a las fallas y acomodar poblaciones temporales de nodos mientras mantienen niveles de conectividad y desempeño aceptables, sin requerir de intermediación o apoyo de una autoridad o servidor centralizado.”

ROADMAP = Hoja de Ruta de un proyecto o ICO, que normalmente se presenta en intervalos de tiempo de Cuartos de Año, es decir Q1, Q2, Q3, Q4 en inglés Quarter.

RPC = Remote Procedure Call, llamada a procedimiento remoto, es un protocolo que permite a los programas llamar a procedimientos localizados en otras máquinas, sin tener que preocuparse por las comunicaciones entre estos. (Sotomayor, 2012)

TOKEN = Ficha…

TRADING = Comerciar con criptomonedas o activos de valor.

WALLET = Billetera Virtual para cada criptomoneda, aquí se utiliza criptografía asimétrica en donde se utilizan las claves pública y privada, la dirección de la Wallet es la clave pública.

Anexo 2. Otros aspectos de las Criptomonedas

Monederos de criptomonedas

Monederos Virtuales

Los monederos virtuales más conocidos en la actualidad son: Jaxx, Bitcoin.com, MyEtherWallet, Coinbase, que son aplicaciones tanto para PC como para dispositivos móviles, cargan una parte de blockchain y tienen seguridades avanzadas como no permitir capturas de pantalla, códigos de apertura, código para envió, etc.

Monederos Físicos

Los monederos físicos o de almacenamiento en frio, (Cold Storage) como Trezzor, Ledger, LedgerNano, etc., éstos monederos pueden estar desconectados de internet y permanecer dentro de una caja fuerte o en un lugar aislado por seguridad, se necesitan las WORD SEED o las palabras semilla para poder desbloquear o recuperar su contenido en otro equipo del mismo tipo.

Monederos de Papel

La importancia y la utilidad de este tipo de monederos, es que están fuera del alcance de spyware y virus que pueden rastrear actividad en computadores infectados. Estos monederos pueden imprimirse en una impresora común y guardarse bajo llave ya que en el mismo documento se encuentra la clave pública y la clave privada por el otro lado, se pueden imprimir en diferentes sitios web, su resultado puede ser similar al siguiente:

Ilustración 39 Generación Aleatoria de Claves de Bitcoin (bitaddress.org, 2017)

Ilustración 40 Cartera de Papel 'PaperWallet' (bitaddress.org, 2018)

Claves Públicas y Privadas de una cuenta:

Ilustración 41 Ejemplo detallado de Claves Pública y Privada de una cartera Bitcoin (bitaddress.org, 2018)

Fuente: recuperado de https://www.bitaddress.org

Nota: Las cuentas de Bitcoin que empiezan por el número “1” son cuentas que requieren una sola clave privada para desbloquear.

1GjG1RTCHxzdjwjZd1YLVwdvjhQj9JoTf8

Las cuentas que empiezan por un número “3” son cuentas conjuntas, de monederos especiales o de corporaciones que necesitan más de una clave para autenticarse, multifirma.

3KgTMRiGFwHzqqdoFC3VCyGfagp9bmD64Q

La red Bitcoin puede generar 2^160 direcciones aceptables de Bitcoin, ‘1.461.501.637.330.902.918.203.684.832.716.283.019.655.932.542.976’, por tal motivo es muy, muy improbable, por no decir imposible, que se pueda generar una clave pública y privada repetidas.

Minería de criptomonedas

La minería de criptomonedas es una de las formas de las cuales se permite asegurar el sistema de cada blockchain con el objetivo de que los datos sean incorruptibles, ya que a mayor número de verificaciones, más difícil será falsificar un dato protegido.

Para explicar brevemente lo que significa la minería en las diferentes criptomonedas, primero se debe comprender lo que es la cadena de bloques (Blockchain) explicada en el punto 2.2 del presente documento.

La minería no es más que la ***verificación o auditoría de las transacciones*** realizadas en cada moneda, el registro en la cadena de bloques, la actualización de un libro contable gigante compartido y sincronizado a nivel mundial por medio del internet.

Minadores

Los Minadores son equipos conectados a la red de cada criptomoneda, que interactúan directamente a la Blockchain confirmando las transacciones realizadas que se encuentran en espera de ser confirmadas y que sus nuevos saldos se registren en el libro mayor.

Software para minería

Dependiendo de la moneda y el algoritmo criptográfico que se pretende minar, existe una gran variedad de software que se puede utilizar, desde los totalmente automatizados que le proporcionan al minador novato una interfaz gráfica para que pueda solamente instalar y minar, pero que a cambio cobra unas comisiones muy altas, hasta los software de código abierto compartidos en redes como GITHUB que demandan un alto conocimiento técnico, que se basan en modo terminal y consola donde se utilizan comandos avanzados, este software no cobra comisión, por tanto los ingresos por utilizar este tipo de minería son más rentables en el largo plazo.

El software más básico y disponible en muchos idiomas, que, además, tiene interfaz gráfica amigable para el usuario básico proviene de la página www.minergate.com, en esta web solo se necesita registro, descargar el software se utiliza el hardware para minar, si no se tiene conocimiento es la opción más recomendable a elegir.

Minergate, posee está conformado por un Pool de minería donde se puede además controlar los equipos (mineros) o trabajadores conectados y el HashPower que ellos generan:

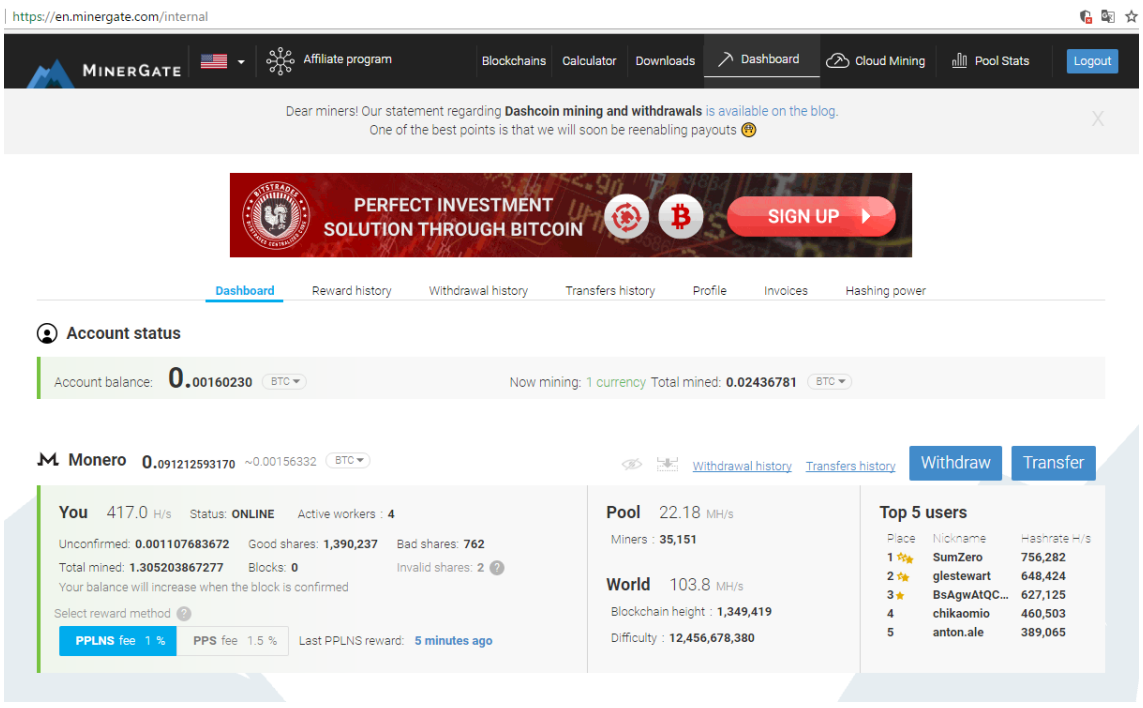


Ilustración 42 Dashboard Minergate (MinerGate, 2018)

Minería Ethereum:

Pools: www.ethermine.org, el software de minado que utiliza GPU tarjetas Gráficas recomienda tanto para Windows como para Linux “Genoils Miner”, trabaja bajo terminal o consola y su comando (escrito en BAT o en SCRIPT) para ejecutar el proceso es el siguiente:

```
setx GPU_FORCE_64BIT_PTR 0
setx GPU_MAX_HEAP_SIZE 100
setx GPU_USE_SYNC_OBJECTS 1
setx GPU_MAX_ALLOC_PERCENT 100
setx GPU_SINGLE_ALLOC_PERCENT 100

ethminer.exe --farm-recheck 200 -G -S eu1.ethermine.org:4444 -FS us1.ethermine.org:4444 -O <Your_Ethereum_Address>.<RigName>
```

Explicación breve del comando:

El primer bloque configura la tarjeta gráfica GPU para que realice operaciones o cálculos matemáticos optimizando el hardware disponible.

El segundo bloque ejecuta el programa con los parámetros insertados a continuación del mismo, PROGRAMA “ethermine.exe”, PARAMETROS “--farm-recheck 200 -G -S eu1.ethermine.org:4444 -FS us1.ethermine.org:4444” configura el pool a donde se va a enviar los resultados de la operación de minería, “-O <Your_Ethereum_Address>. <RigName>” -O es el Usuario donde se procederá a ingresar la dirección de una billetera ETHEREUM a donde quiere que se le realicen los pagos por el trabajo de minería realizado. El RIG_NAME es el nombre o Nick que le damos a ese especifico trabajador, para poderlo identificar en caso de tener más de uno.

Otro software que se utiliza para minado de Ethereum es:

Claymore Miner, que se puede obtener de la página: https://github.com/nanopool/Claymore-Dual-Miner/releases

Ethminer, que se puede obtener de la web: https://ethminer.io/

Qtminer, que se obtiene de la página: https://github.com/etherchain-org/qtminer?files=1

El siguiente es un registro del trabajo de los equipos de minería en la página www.ethermine.org, en el cual se registra los equipos que se encuentran minando, la cantidad de hashrate, los shares enviados a la pool, el grafico del historial, etc.

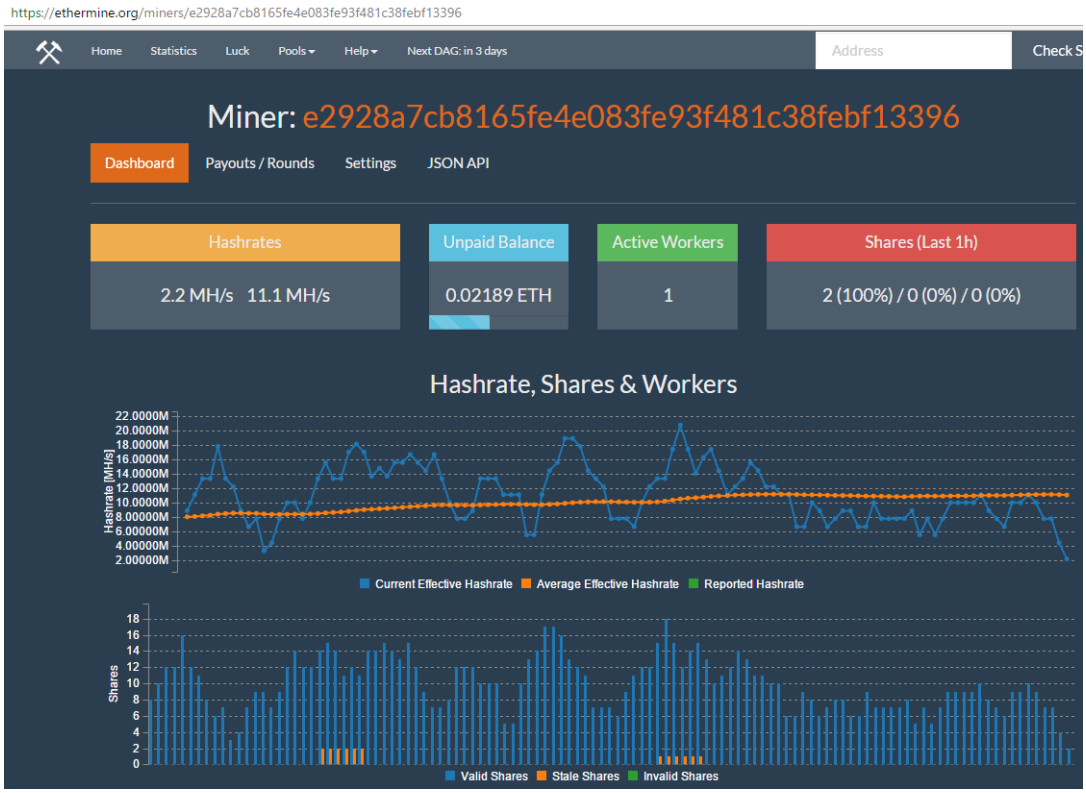


Ilustración 43 Dashboard Ethermine.org (Ethermine, 2018)

El minado de criptomonedas se divide en minado por CPU, minado por GPU y minado con equipos especializados ASIC.

Se puede aún realizar minado por CPU al algoritmo CRYPTONIGTH, en el mismo que constan las siguientes monedas:

Name(Tag) Algorithm	Block Time Block Reward Last Block	Difficulty NetHash	Est. Rewards Est. Rewards 24h	Exchange Rate	Market Cap Volume	Rev. BTC Rev. 24h	Rev. $ Profit	Profitability Current | 24h 3 days | 7 days
Monero(XMR) CryptoNightV7	BT: 1m 51s BR: 3.96 LB: 1,651,877	**59,900,814,933** 539.65 Mh/s 4.2%	0.0983 0.1025	0.01702800 (HitBTC) 1.0%	$2,002,278,633 **2,495.13 BTC**	0.00167 0.00175	$12.54 **$7.02**	91% | 96% 97% | **100%**
Graft(GRFT) CryptoNightV7	BT: 1m 57s BR: 1,414.86 LB: 164,184	**1,775,061,861** 15.17 Mh/s -7.2%	1,183.1748 1,099.7926	0.00000158 (Cryptopia) 1.8%	$4,288,336 **1.30 BTC**	0.00187 0.00174	$12.48 **$6.96**	**102%** | 95% 96% | 98%
Nicehash-CNV7 CryptoNightV7	BT: - BR: - LB: -	- 63.41 Mh/s -7.0%	0.00164 0.00169	0.09560000 (Nicehash) -2.8%	- **6.70 BTC**	0.00164 0.00169	$12.16 **$6.64**	89% | **93%** 87% | 86%
DigitalNote(XDN) CryptoNightV7	BT: 2m 5s BR: 150.00 LB: 713,135	**385,722,685** 3.09 Mh/s 3.8%	574.7040 596.4693	0.00000052 (Bittrex) 6.0%	$25,877,120 **153.54 BTC**	0.00030 0.00031	$2.23 **-$3.29**	16% | 17% 18% | **19%**

Ilustración 44 Monedas de Algoritmo CryptoNight (What to mine, 2018)

El software de minado para este algoritmo es XMR-STACK.EXE

Este software apunta al pool de Monero = www.minexmr.com

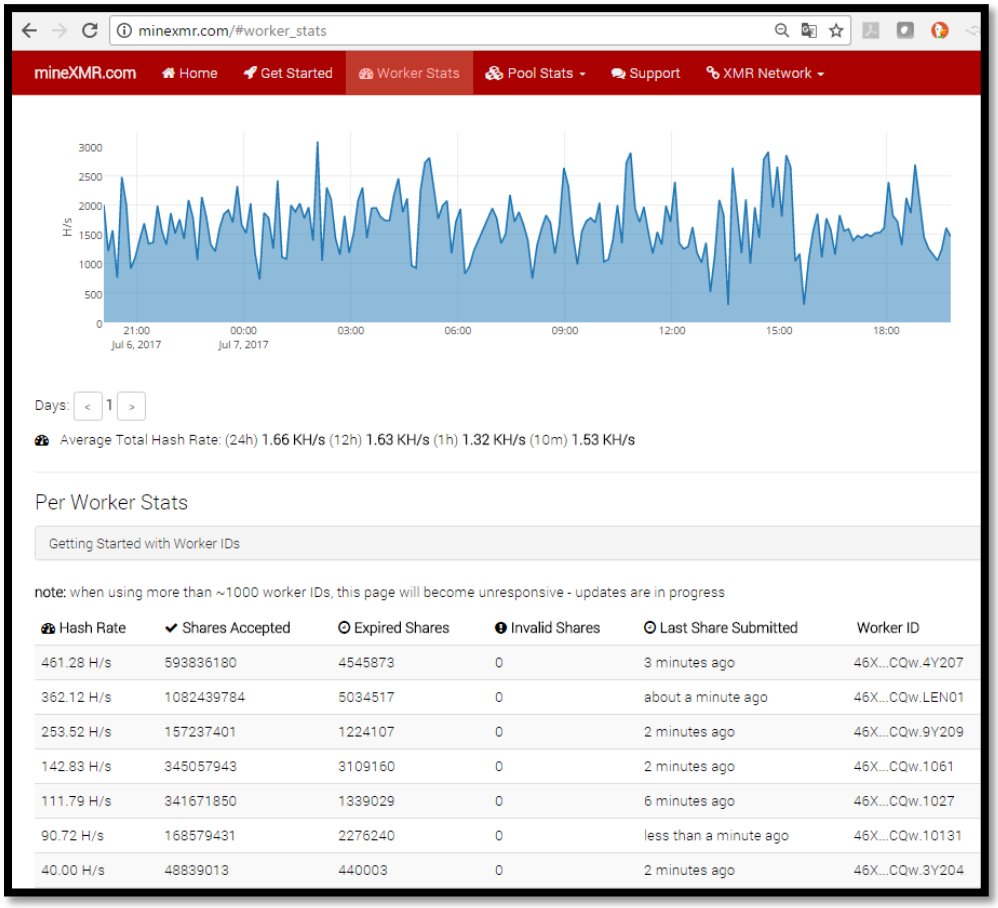


Ilustración 45 Dashboard MINEXMR (mineXMR.com, 2018)

Las tarjetas gráficas GPU más utilizadas en la actualidad (que siguen siendo rentables):

AMD ATI Radeon – modelos:

RX 280, RX 290, RX 380, RX 390

RX 470, RX 480. RX 550

RX 570, RX 580, RX FURY

NVIDIA – modelos:

GTX 960, GTX 970, GTX980, GTX 980 Ti

GTX 750Ti, GTX 1050 Ti, GTX 1060,

GTX 1070, GTX 1080, GTX 1080Ti

Todas estas tarjetas de video utilizadas para video juegos y diseño gráfico, al momento se encuentran agotadas y si se encuentran en el mercado, constan con precios sumamente elevados por la gran demanda de la minería en la actualidad está irrumpiendo este mercado aceleradamente.

NVIDIA y AMD al ver la gran demanda de sus equipos con otro fin, se decidieron fabricar tarjetas especializadas para minería que no poseen salida de video, solamente los procesadores, estas tarjetas se proyectan que saldrán al mercado en el mes de julio / agosto de 2017.

Ilustración 46 Tarjeta Video sin salida de video (Hollingworth, 2017)

Equipos ASIC

ASIC (Application Specific Integrated Circuit), Circuito Integrado de Uso Específico, es decir, es un computador hecho a medida para trabajos específicos, en este caso podríamos decir que un robot o equipo que coloca los parabrisas en una línea de producción de vehículos es un ASIC, ya que se encuentra diseñado para realizar solamente esa función, los equipos de minería ASIC, tienen la función específica de calcular los algoritmos específicos para los que pueden ser creados, estos equipos tienen un periodo de vida relativamente corto, 12 a 18 meses, luego de este período dejan de ser rentables.

Se pueden minar las siguientes criptomonedas que utilizan Algoritmo SHA256:

Bitcoin, Bitcoin Cash, Peercoin, DGB-SHA, Zcoin, Crown, etc.

Los equipos más conocidos son fabricados por la empresa BITMAIN con su modelo, “Antminer S9”, que tiene un costo aproximado de $1000 usd, el mismo que permanece encendido las 24 horas del día, genera un promedio de $200 usd mensuales, tiene un consumo de energía eléctrica de 1400 vatios / hora. Su hashrate es de 13.5 Tera Hashes (13.500 GH/s) para el algoritmo SHA256.

Ilustración 47 Antminer S9 (BITMAIN, 2018)

Mal uso de las criptomonedas

El uso del Bitcoin y las criptomonedas, no es sinónimo de ciber-delincuencia, fraude o estafa, porque como todas las cosas o artefactos que se han creado, dependen del uso que se les dé, todo sirve tanto para hacer el bien como para hacer el mal, depende de las personas que lo utilicen, tanto como un cuchillo sirve para alimentar, como para asesinar.

Fraude piramidal, en base a la gran popularidad que ha tomado el Bitcoin y el desconocimiento general de su funcionamiento, es un buen producto para la realización de estafas y en muchos casos lo disfrazan bajo esquemas Ponzi con propaganda de Bitcoin o criptomonedas, casos recientes publicados en periódico de Ecuador (PubliMetro, 2017): ONECOIN - MECOIN, MYTRADERCOIN, GLADIACOIN, etc., han defraudado cientos de miles de personas perjudicándolos en una gran cantidad de dinero, se debe tomar en cuenta que el dinero que se piensa invertir en criptodivisas, es un dinero de muy alto riesgo, no deben ser utilizados valores que se necesitan para pagar créditos, vivienda, alimentación, servicios básicos, etc.

En la publicación de (Flores, 2017) en Bolivia un SCAM con un nombre homónimo al de la real criptomoneda “Bitcoin Cash” estafó a más de 100.000 personas.

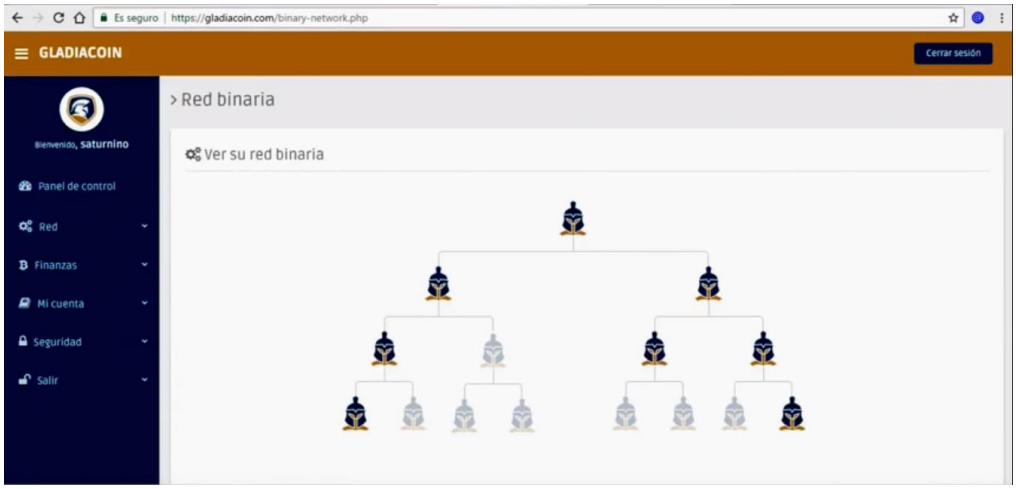


Ilustración 48 Esquema Ponzi Página GladiaCoin (hoekomikaangeld.com, 2017)

Lavado de Dinero. En este caso existen muchos usuarios que por el hecho de que las criptomonedas no se encuentran regidas por sistemas financieros o por gobiernos las utilizan para hacer el blanqueo de activos o lavado de dinero, que es un acto no legal, pero además se han creado monedas como MONERO que es imposible rastrear, solamente las transacciones reflejan para las personas envueltas en la transacción, es decir el que recibe, tiene un mensaje en su Wallet que dice: “Remitente Anónimo”

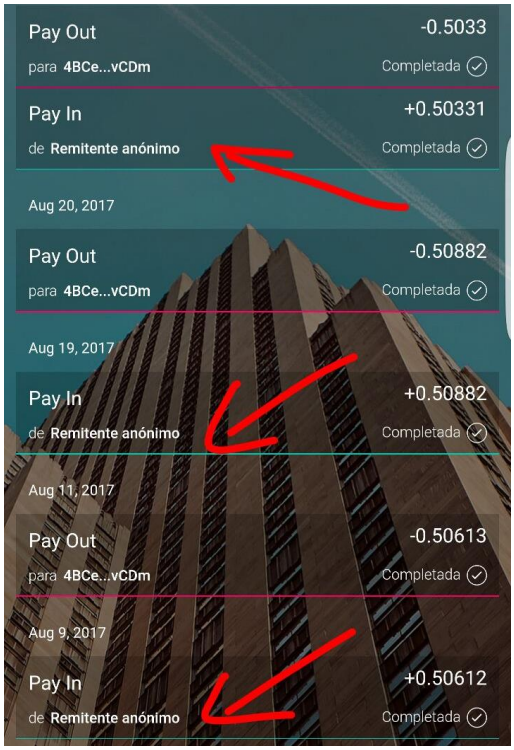


Ilustración 49 Transacciones de Monero señala 'Remitente Anónimo'

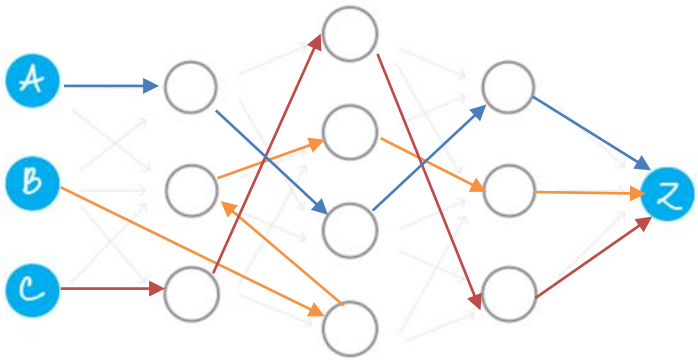


Ilustración 50 Transacción No Rastreable Monero

Deep-Web / Silk Road

El manejo de Bitcoin se popularizó en la Deep web para esconder los pagos de compras de cosas ilegales, como armas, pornografía, drogas, incluso se utilizaba para la compra y contratación de ataques tanto físicos como ciber-ataques, compras de vulnerabilidades de Zero Day que eran pagadas al triple del valor que la misma empresa dueña del software podría pagar.

Como nos describe (Rojas, 2015) Silk-Road 'Ruta de la Seda' era una tienda online implementada bajo la red Thor y los sitios Onion que se hacía muy difícil seguir su rastro y las autoridades no podían aun encontrar quien era el famoso "Pyrate Roberts" Ross Ulbritch, hasta que cometió un error y fue encontrado en una biblioteca y arrestado y la tienda online detuvo sus servicios que mantenía activos desde enero de 2011 hasta octubre de 2013.

Monedas de proyectos muertos

Existe gran cantidad de proyectos que han sido abandonados por no tener un objetivo convincente o útil a la comunidad de criptomonedas, a las mismas se hace referencia en un sitio web donde se publican las criptomonedas o proyectos que no tuvieron el efecto deseado, dejaron de ser apoyadas e incluso fueron estafas o scam, http://deadcoins.com/.

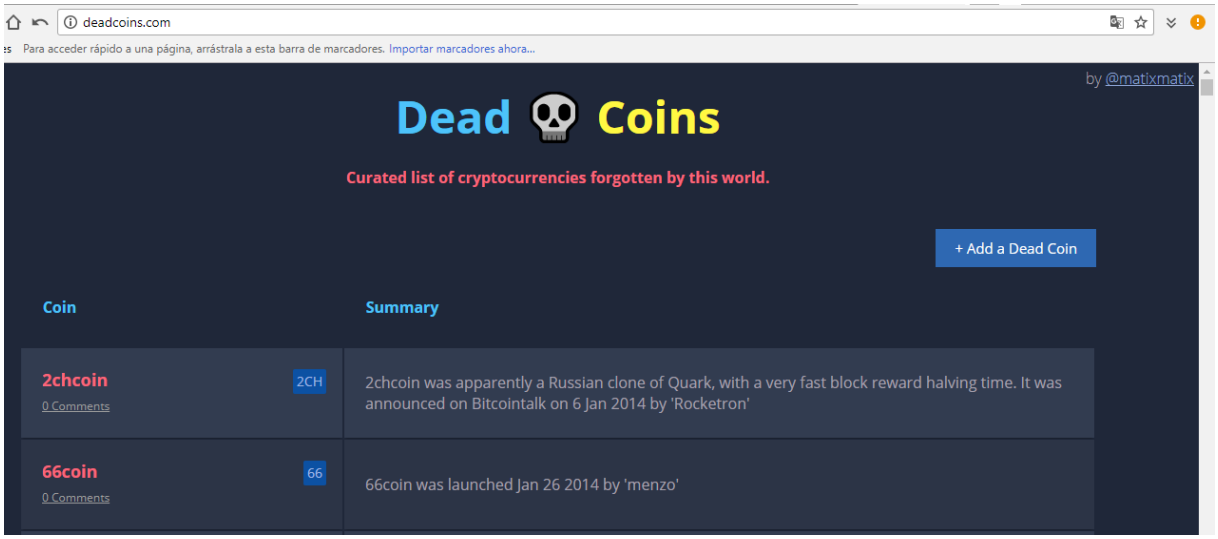


Ilustración 51 Sitio Web de Monedas Muertas (Dead Coins, 2018)

Anexo 3. Proyectos que aceptan SmartContracts

Lisk (LSK)

Lisk es la criptomoneda creada en el 2016 que permite la programación de contratos inteligentes directamente sobre lenguaje JavaScript y permite la implantación de cadenas laterales.

Neo (NEO)

Neo es denominada también el Ethereum chino, permite la programación de contratos inteligentes sobre su red, en sus inicios tuvo una gran acogida subiendo en su popularidad dentro de las 10 primeras criptomonedas por el potencial que ofrecía, en la actualidad se han filtrado algunos fallos de forma en su proyecto que ha hecho que el valor de su criptomoneda disminuya en un 80%.

Cardano (ADA)

Una propuesta avanzada que pretende ser un proyecto encaminado a ser superior a Ethereum, ha tenido una gran acogida en el ambiente de los criptoactivos y sistemas para funcionamiento de los contratos inteligentes, está catalogada como un criptoativo de nivel 3, tomando en cuenta que Bitcoin seria nivel 1 y Ethereum nivel 2. Lo que diferencia a Cardano de los demás proyectos es la creación de una arquitectura de seguridad basada en capas, que utiliza un lenguaje llamado **Haskell**.

Anexo 4. Convenciones para contratos inteligentes

Guía de estilo

Introducción

Esta guía pretende proporcionar convenciones de codificación para escribir código con Solidity. Esta guía debe ser entendida como un documento en evolución que cambiará con el tiempo según aparecen nuevas convenciones útiles y antiguas convenciones se vuelven obsoletas.

Muchos proyectos implementarán sus propias guías de estilo. En el caso de conflictos, las guías de estilo específicas del proyecto tendrán prioridad.

La estructura y muchas de las recomendaciones de esta guía de estilo fueron tomadas de Python: guía de estilo pep8.

El objetivo de esta guía no es ser la forma correcta o la mejor manera de escribir código con Solidity. El objetivo de esta guía es la consistencia. Una cita de python pep8 capta bien este concepto.

Una guía de estilo es sobre consistencia. La consistencia con esta guía de estilo es importante. La consistencia dentro de un proyecto es más importante. La consistencia dentro de un módulo o función es lo más importante. Pero sobre todo: saber cuándo ser inconsistente - a veces la guía de estilo simplemente no se aplica. En caso de duda, use su mejor juicio. Mire otros ejemplos y decida qué parece mejor. ¡Y no dude en preguntar!

Diseño del código

Sangría

Utilice 4 espacios por nivel de sangría.

Tabulador o espacios

Los espacios son el método de sangría preferido.

Se deben evitar la mezcla del tabulador y los espacios.

Líneas en blanco

Envuelva las declaraciones de nivel superior en el código de Solidity con dos líneas en blanco.

Sí:

```
contract A {

    ...

}




contract B {

    ...

}




contract C {

    ...

}
```

No:

```
contract A {

    ...

}

contract B {

    ...

}


contract C {

    ...

}
```

Dentro de un contrato, rodee las declaraciones de una función con una sola línea en blanco.

Las líneas en blanco se pueden omitir entre grupos de una frase relacionada (tales como las funciones stub en un contrato abstracto)

Sí:

```
contract A {

    function spam();

    function ham();

}




contract B is A {

    function spam() {

        ...

    }


    function ham() {

        ...

    }

}
```

No:

```
contract A {

    function spam() {

        ...

    }

    function ham() {

        ...

    }

}
```

Codificación de archivos de origen

Se prefiere la codificación del texto en UTF-8 o ASCII.

Importación

Las declaraciones de importación siempre deben colocarse en la parte superior del archivo.

Sí:

```
import "owned";




contract A {
```

```
    ...
}
```

```
contract B is owned {
    ...
}
```

No:

```
contract A {
    ...
}
```

```
import "owned";
```

```
contract B is owned {
    ...
}
```

Orden de funciones

La ordenación ayuda a que los lectores puedan identificar las funciones que pueden invocar y encontrar las definiciones de constructor y de retorno más fácilmente.

Las funciones deben agruparse de acuerdo con su visibilidad y ser ordenadas de acuerdo a:

constructor

función fallback (Si existe)

external

public

internal

private

Dentro de un grupo, coloque las funciones constant al final.

Sí:

```
contract A {
    function A() {
        ...
    }

    function() {
```

```
        ...
    }

    // Funciones external
    // ...

    // Funciones external que son constantes
    // ...

    // Funciones public
    // ...

    // Funciones internal
    // ...

    // Funciones private
    // ...
}
```

No:

```
contract A {

    // Funciones external
    // ...

    // Funciones private
    // ...

    // Funciones public
    // ...

    function A() {
        ...
    }

    function() {
        ...
```

```
    }


    // Funciones internal

    // ...

}
```

Espacios en blanco en expresiones

Evite los espacios en blanco superfluos en las siguientes situaciones:

Inmediatamente entre paréntesis, llaves o corchetes, con la excepción de declaraciones de una función en una sola línea.

Sí:

```
spam(ham[1], Coin({name: "ham"}));
```

No:

```
spam( ham[ 1 ], Coin( { name: "ham" } ) );
```

Excepción:

```
function singleLine() { spam(); }
```

Inmediatamente antes de una coma, punto y coma:

Sí:

```
function spam(uint i, Coin coin);
```

No:

```
function spam(uint i , Coin coin) ;
```

Más de un espacio alrededor de una asignación u otro operador para alinearlo con otro:

Sí:

```
x = 1;

y = 2;

long_variable = 3;
```

No:

```
x             = 1;

y             = 2;

long_variable = 3;
```

No incluya un espacio en blanco en la función fallback:

Sí:

```
function() {

    ...

}
```

No:

```
function () {

    ...
```

```
}
```

Estructuras de control

Las llaves que denotan el cuerpo de un contrato, biblioteca, funciones y estructuras deberán:

Abrir en la misma línea que la declaración

Cerrar en su propia línea en el mismo nivel de sangría que la declaración.

La llave de apertura debe ser procedida por un solo espacio.

Sí:

```
contract Coin {
    struct Bank {
        address owner;
        uint balance;
    }
}
```

No:

```
contract Coin
{
    struct Bank {
        address owner;
        uint balance;
    }
}
```

Las mismas recomendaciones se aplican a las estructuras de control if, else, while y for.

Además, debería existir un único espacio entre las estructuras de control if, while, y for, y el bloque entre paréntesis que representa el condicional, así como un único espacio entre el bloque del paréntesis condicional y la llave de apertura.

Sí:

```
if (...) {
    ...
}


for (...) {
    ...
}
```

No:

```
if (...)
{
    ...
```

```
}


while(...){
}


for (...) {
    ...;}
```

Para las estructuras de control cuyo cuerpo sólo contiene declaraciones únicas, se pueden omitir los corchetes si la declaración cabe en una sola línea.

Sí:

```
if (x < 10)
    x += 1;
```

No:

```
if (x < 10)
    someArray.push(Coin({
        name: 'spam',
        value: 42
    }));
```

Para los bloques if que contienen una condición else o else if, el else debe estar en la misma línea que el corchete de cierre del if. Esto es una excepción en comparación con las reglas de otras estructuras de tipo bloque.

Sí:

```
if (x < 3) {
    x += 1;
} else if (x > 7) {
    x -= 1;
} else {
    x = 5;
}



if (x < 3)
    x += 1;
else
    x -= 1;
```

No:

```
if (x < 3) {
    x += 1;
```

```
}

else {

    x -= 1;

}
```

Declaración de funciones

Para declaraciones de función cortas, se recomienda dejar el corchete de apertura del cuerpo de la función en la misma línea que la declaración de la función.

El corchete de cierre debe estar al mismo nivel de sangría que la declaración de la función.

El corchete de apertura debe estar precedido por un solo espacio.

Sí:

```
function increment(uint x) returns (uint) {

    return x + 1;

}



function increment(uint x) public onlyowner returns (uint) {

    return x + 1;

}
```

No:

```
function increment(uint x) returns (uint)

{

    return x + 1;

}



function increment(uint x) returns (uint){

    return x + 1;

}



function increment(uint x) returns (uint) {

    return x + 1;

    }



function increment(uint x) returns (uint) {

    return x + 1;}
```

Se debe especificar la visibilidad de los modificadores para una función antes de cualquier modificador personalizado.

Sí:

```
function kill() public onlyowner {
```

```
    selfdestruct(owner);

}
```

No:

```
function kill() onlyowner public {

    selfdestruct(owner);

}
```

Para las declaraciones de función largas, se recomienda dejar a cada argumento su propia línea al mismo nivel de sangría que el cuerpo de la función. El paréntesis de cierre y el corchete de apertura deben de estar en su propia línea también y con el mismo nivel de sangría que la declaración de la función.

Sí:

```
function thisFunctionHasLotsOfArguments(

    address a,

    address b,

    address c,

    address d,

    address e,

    address f

) {

    doSomething();

}
```

No:

```
function thisFunctionHasLotsOfArguments(address a, address b, address c,

    address d, address e, address f) {

    doSomething();

}


function thisFunctionHasLotsOfArguments(address a,

                                        address b,

                                        address c,

                                        address d,

                                        address e,

                                        address f) {

    doSomething();

}


function thisFunctionHasLotsOfArguments(

    address a,
```

```
    address b,
    address c,
    address d,
    address e,
    address f) {
    doSomething();
}
```

Si una declaración de función larga tiene modificadores, cada uno de ellos debe de estar en su propia línea.

Sí:

```
function thisFunctionNameIsReallyLong(address x, address y, address z)
    public
    onlyowner
    priced
    returns (address)
{
    doSomething();
}


function thisFunctionNameIsReallyLong(
    address x,
    address y,
    address z,
)
    public
    onlyowner
    priced
    returns (address)
{
    doSomething();
}
```

No:

```
function thisFunctionNameIsReallyLong(address x, address y, address z)
                                      public
                                      onlyowner
                                      priced
                                      returns (address) {
```

```
    doSomething();
}

function thisFunctionNameIsReallyLong(address x, address y, address z)
    public onlyowner priced returns (address)
{
    doSomething();
}

function thisFunctionNameIsReallyLong(address x, address y, address z)
    public
    onlyowner
    priced
    returns (address) {
    doSomething();
}
```

Para las funciones de tipo constructor en contratos heredados que requieren argumentos, si la declaración de la función es larga o difícil de leer, se recomienda poner cada constructor base en su propia línea de la misma manera que con los modificadores.

Sí:

```
contract A is B, C, D {
    function A(uint param1, uint param2, uint param3, uint param4, uint param5)
        B(param1)
        C(param2, param3)
        D(param4)
    {
        // do something with param5
    }
}
```

No:

```
contract A is B, C, D {
    function A(uint param1, uint param2, uint param3, uint param4, uint param5)
    B(param1)
    C(param2, param3)
    D(param4)
    {
        // do something with param5
```

```
    }
}


contract A is B, C, D {
    function A(uint param1, uint param2, uint param3, uint param4, uint param5)
        B(param1)
        C(param2, param3)
        D(param4) {
        // do something with param5
    }
}
```

Cuando se declara funciones cortas con una sola declaración, está permitido hacerlo en una solo línea.

Permisible:

```
function shortFunction() { doSomething(); }
```

Esta guía sobre la declaración de funciones está pensada para mejorar la legibilidad. Sin embargo, los autores deberían utilizar su mejor juicio, ya que esta guía tampoco intenta cubrir todas las posibles permutaciones para las declaraciones de función.

Declaración de variables

La declaración de variables tipo array no debe incluir un espacio entre el tipo y el corchete.

Sí:

```
uint[] x;
```

No:

```
uint [] x;
```

Otras recommendaciones

Los strings deben de citarse con comillas dobles en lugar de comillas simples.

Sí:

```
str = "foo";

str = "Hamlet says, 'To be or not to be...'";
```

No:

```
str = 'bar';

str = '"Be yourself; everyone else is already taken." -Oscar Wilde';
```

Se envuelve los operadores con un solo espacio de cada lado.

Sí:

```
x = 3;

x = 100 / 10;

x += 3 + 4;

x |= y && z;
```

No:

```
x=3;

x = 100/10;

x += 3+4;

x |= y&&z;
```

Para los operadores con una prioridad mayor que otros, se pueden omitir los espacios de cada lado del operador para marcar la precedencia. Esto se hace para mejorar la legibilidad de declaraciones complejas. Se debe usar siempre el mismo número de espacios a cada lado de un operador.

Sí:

```
x = 2**3 + 5;

x = 2*y + 3*z;

x = (a+b) * (a-b);
```

No:

```
x = 2** 3 + 5;

x = y+z;

x +=1;
```

Convención sobre nombres

Las convenciones sobre nombres son extremadamente útiles siempre y cuando se usen de forma ámplia. El uso de diferentes convenciones puede transmitir meta información significativa a la que de otro modo no tendríamos acceso de manera inmediata.

Las recomendaciones de nombres que se dan aquí están pensadas para mejorar la legibilidad, y por lo tanto no se deben considerar como reglas. Son más bien una guía para intentar transmitir la mayor información posible a través del nombre de las cosas.

Finalmente, la consistencia dentro de un bloque de código siempre debe prevalecer sobre cualquier convención destacada en este documento.

Estilos para poner nombres

Para evitar confusiones, se usarán los siguiente nombres para referirse a diferentes estilos para poner nombres.

b (letra minúscula única)

B (letra mayúscula única)

minuscula

minuscula_con_guiones_bajos

MAYUSCULA

MAYUSCULA_CON_GUIONES_BAJOS

PalabrasConLaInicialEnMayuscula (también llamado CapWords)

mezclaEntreMinusculaYMayuscula (¡distinto a PalabrasConLaInicialEnMayuscula por el uso de una minúscula en la letra inicial!)

Palabras_Con_La_Inicial_En_Mayuscula_Y_Guiones_Bajos

Nota

Cuando se usan abreviaciones en CapWords, usar mayúsculas para todas las letras de la abreviación. Es decir que HTTPServerError es mejor que HttpServerError

Nombres a evitar

l - Letra minúscula ele

O - Letra mayúscula o

I - Letra mayúscula i

No usar jamás ninguna de estas letras únicas para nombrar una variable. Estas letras generalmente no se diferencian de los dígitos uno y cero.

Contratos y librerías de nombres

Contratos y librerías deben de nombrarse usando el estilo CapWord (PalabrasConLaInicialEnMayuscula).

Eventos

Los eventos deben de nombrarse usando el estilo CapWord (PalabrasConLaInicialEnMayuscula).

Nombres para funciones

Las funciones deben de nombrarse usando el estilo mezclaEntreMinusculaYMayuscula.

Argumentos de funciones

Cuando se escriben funciones de librerías que operan sobre un struct personalizado, el struct debe ser el primer argumento y debe nombrarse siempre self.

Variables locales y de estado

Usar el estilo mezclaEntreMinusculaYMayuscula.

Constantes

Las constantes deben de nombrarse con todas las letras mayúsculas y guiones bajos para separar las palabras (p.ej. MAX_BLOCKS).

Modificadores

Usar el estilo mezclaEntreMinusculaYMayuscula.

Evitar Colisiones

guion_bajo_unico_con_cola_

Se recomienda usar esta convención cuando el nombre deseado colisiona con un nombre inherente al sistema o reservado.

Anexo 5. Contrato Básico de Token

```
1.  pragma solidity ^0.4.20;
2.
3.  contract MyToken {
4.  /* Esto crea un array con todos los saldos */
```

```
5.  mapping (address => uint256) public balanceOf;
6.
7.  /* Inicializa el contrato con el balance inicial de los Tokens al creador del contrato */
8.  function MyToken(
9.      uint256 initialSupply
10.     ) public {
11.     balanceOf[msg.sender] = initialSupply;          // Da al creador el saldo inicial
12. }
13.
14. /* Enviar Monedas */
15. function transfer(address _to, uint256 _value) public returns (bool success) {
16.     require(balanceOf[msg.sender] >= _value);     // Verifica el saldo del remitente
17.     require(balanceOf[_to] + _value >= balanceOf[_to]); // Verifica sobreflujo
18.     balanceOf[msg.sender] -= _value;                    // Retira del remitente
19.     balanceOf[_to] += _value;                           // Agrega al receptor
20.     return true;
21.     }
22. }
```